

Sintaxis y procesamiento de cifrado XML

Recomendación del W3C del 10 de diciembre de 2002

Esta versión:
http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
Ultima versión:
http://www.w3.org/TR/xmlenc-core/
Versión previa:
http://www.w3.org/TR/2002/PR-xmlenc-core-20021003/

Editores
Donald Eastlake <dee3@torque.pothole.com>
José Reagle <reagle@w3.org>
Autores
Takeshi Imamura <IMAMU@jp.ibm.com>
Blair Dillaway <blaird@microsoft.com>
Ed Simón <edsimon@xmlsec.com>
Colaboradores
Ver participantes .

Consulte las **erratas** de este documento, que pueden incluir algunas correcciones normativas. Ver también **traducciones** .

Abstracto

Este documento especifica un proceso para cifrar datos y representar el resultado en XML. Los datos pueden ser datos arbitrarios (incluido un documento XML), un elemento XML o contenido de un elemento XML. El resultado del cifrado de datos es un elemento de cifrado XML que contiene o hace referencia a los datos cifrados.

Estado de este documento

Este documento es la Recomendación de cifrado XML (REC) del W3C . Este documento ha sido revisado por miembros del W3C y otras partes interesadas y ha sido respaldado por el Director como Recomendación del W3C. Es un documento estable y puede usarse como material de referencia o citarse como referencia normativa de otro documento. El papel del W3C al elaborar la Recomendación es llamar la atención sobre la especificación y promover su implementación generalizada. Esto mejora la funcionalidad y la interoperabilidad de la Web.

Esta especificación fue producida por el Grupo de Trabajo de Cifrado XML del W3C (Actividad), que cree que la especificación es suficiente para la creación de implementaciones interoperables independientes como se demuestra en el Informe de Interoperabilidad.

Las divulgaciones de patentes relevantes para esta especificación se pueden encontrar en la página de divulgación de patentes del Grupo de Trabajo de conformidad con la política del W3C.

Informe los errores en este documento a xml-encryption@w3.org (archivo público).

La lista de errores conocidos en esta especificación está disponible en http://www.w3.org/Encryption/2002/12-xmlenc-errata .

La versión en inglés de esta especificación es la única versión normativa. La información sobre las traducciones de este documento (si corresponde) está disponible en http://www.w3.org/Encryption/2002/12-xmlenc-translations .

Puede encontrar una lista de las recomendaciones actuales del W3C y otros documentos técnicos en http://www.w3.org/TR/ .

Tabla de contenido

---

1 Introducción

Este documento especifica un proceso para cifrar datos y representar el resultado en XML. Los datos pueden ser datos arbitrarios (incluido un documento XML), un elemento XML o contenido de un elemento XML. El resultado del cifrado de datos es un elemento de cifrado XML `EncryptedData`que contiene (a través de uno de sus contenidos secundarios) o identifica (a través de una referencia URI) los datos cifrados.

Al cifrar un elemento XML o el contenido del elemento, el `EncryptedData`elemento reemplaza el elemento o el contenido (respectivamente) en la versión cifrada del documento XML.

Al cifrar datos arbitrarios (incluidos documentos XML completos), el `EncryptedData`elemento puede convertirse en la raíz de un nuevo documento XML o convertirse en un elemento secundario en un documento XML elegido por la aplicación.

1.1 Convenciones editoriales y de conformidad

Esta especificación utiliza esquemas XML [esquema XML] para describir el modelo de contenido.

Las palabras clave "DEBE", "NO DEBE", "REQUERIDO", "DEBE", "NO DEBE", "DEBE", "NO DEBE", "RECOMENDADO", "PUEDE" y "OPCIONAL" en esta especificación son debe interpretarse como se describe en RFC2119 [PALABRAS CLAVE]:

> "Sólo DEBEN utilizarse cuando realmente sea necesario para la interoperación o para limitar un comportamiento que pueda causar daño (por ejemplo, limitar las retransmisiones)"

En consecuencia, utilizamos estas palabras clave en mayúscula para especificar sin ambigüedades los requisitos sobre las características y el comportamiento del protocolo y la aplicación que afectan la interoperabilidad y la seguridad de las implementaciones. Estas palabras clave no se utilizan (en mayúscula) para describir la gramática XML; Las definiciones de esquema describen inequívocamente dichos requisitos y deseamos reservar la prominencia de estos términos para las descripciones en lenguaje natural de protocolos y características. Por ejemplo, un atributo XML podría describirse como "opcional". El cumplimiento de la especificación del espacio de nombres XML [XML-NS] se describe como "REQUERIDO".

1.2 Filosofía del diseño

La filosofía de diseño y los requisitos de esta especificación (incluidas las limitaciones relacionadas con la validez de la instancia) se abordan en los Requisitos de cifrado XML [EncReq].

1.3 Versiones , espacios de nombres, URI e identificadores

No se prevé ningún número de versión explícito en esta sintaxis. Si se necesita una versión futura, utilizará un espacio de nombres diferente. El URI del espacio de nombres XML experimental [XML-NS] que DEBEN utilizar las implementaciones de esta especificación (anticuada) es:

```
xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
```

Este espacio de nombres también se utiliza como prefijo para los identificadores de algoritmos utilizados por esta especificación. Si bien las aplicaciones DEBEN admitir XML y espacios de nombres XML, el uso de entidades internas [XML , sección 4.2.1], el " " prefijo del espacio de nombres`xenc` XML [XML-NS , sección 2] y las convenciones de configuración predeterminada/alcance son OPCIONALES; Utilizamos estas funciones para proporcionar ejemplos compactos y legibles. Además, la entidad se define para proporcionar identificadores abreviados para los URI definidos en esta especificación. Por ejemplo " corresponde a "http://www.w3.org/2001/04/xmlenc#Element".`&xenc;&xenc;Element"`

Esta especificación utiliza el espacio de nombres y las definiciones de esquema de la firma XML [XML-DSIG].

```
xmlns:ds='http://www.w3.org/2000/09/xmldsig#'
```

Los URI [URI] DEBEN cumplir con la definición de tipo [XML-Schema] y la especificación [XML-DSIG , 4.3.3.1 El atributo URI] (es decir, caracteres permitidos, escape de caracteres, soporte de esquema, etc.).`anyURI`

1.4 Agradecimientos

Se agradecen las contribuciones de los siguientes miembros del Grupo de Trabajo a esta especificación de acuerdo con las políticas de contribuyentes y la lista activa del Grupo de Trabajo .

- joseph ashwood
- Simon Blake-Wilson, Certicom
- Frank D. Cavallito, Sistemas BEA
- Eric Cohen, PricewaterhouseCoopers
- Blair Dillaway, Microsoft (Autor)
- Blake Dournaee, Seguridad RSA
- Donald Eastlake, Motorola (Editor)
- Barb Fox, Microsoft
- Christian Geuer-Pollmann, Universidad de Siegen
- Tom Gindin, IBM
- Jiandong Guo, Faos
- Phillip Hallam-Baker, Verisign
- Amir Herzberg, NewGenPay
- Merlín Hughes, Baltimore
- Federico Hirsch
- Maryann Hondo, IBM
- Takeshi Imamura, IBM (Autor)
- Mike Just, Entrust, Inc.
- Brian La Macchia, Microsoft
- Hiroshi Maruyama, IBM
- John Messing, Ley en línea
- Shivaram Mysore, Sun Microsystems
- Thane Plambeck, Verisign
- Joseph Reagle, W3C (Presidente, Editor)
- Alexei Sanin
- Jim Schaad, consultoría Soaring Hawk
- Ed Simon, XMLsec (Autor)
- Daniel Toth, Ford
- Yongge Wang, Certicom
- Steve Wiley, mi prueba

Además, agradecemos a las siguientes personas por sus comentarios durante y después de la última llamada:

- Martín Durst, W3C
- Dan Lanz, Zolera
- Susan Lesch, W3C
- David Orchard, Sistemas BEA
- Ronald Rivest, MIT

2 Descripción general y ejemplos de cifrado (no normativo)

Esta sección proporciona una descripción general y ejemplos de la sintaxis de cifrado XML. La sintaxis formal se encuentra en *Sintaxis de cifrado* (sección 3); el procesamiento específico se proporciona en *las Reglas de procesamiento* (sección 4).

Expresado en forma abreviada, el `EncryptedData`elemento tiene la siguiente estructura (donde "?" denota cero o una ocurrencia; "+" denota una o más ocurrencias; "*" denota cero o más ocurrencias; y la etiqueta de elemento vacía significa que el elemento debe ser vacío):

```
<¿Identificación de datos cifrados? ¿Tipo? ¿Tipo de Mimica? ¿Codificación?>
  <Método de cifrado/>?
  <ds:Información clave>
    <Clave cifrada>?
    <Método de acuerdo>?
    <ds:NombreClave>?
    <ds:Método de recuperación>?
    <ds:*>?
  </ds:KeyInfo>?
  <Datos cifrados>
    <ValorCifrado>?
    <¿URI de referencia de cifrado?>?
  </CipherData>
  <Propiedades de cifrado>?
</EncryptedData>
```

El `CipherData`elemento envuelve o hace referencia a los datos cifrados sin procesar. Si es envolvente, los datos cifrados sin procesar son el `CipherValue`contenido del elemento; si se hace referencia, el atributo `CipherReference`del elemento `URI`apunta a la ubicación de los datos cifrados sin procesar

2.1 Granularidad del cifrado

Considere la siguiente información de pago ficticia, que incluye información de identificación e información apropiada para un método de pago (por ejemplo, tarjeta de crédito, transferencia de dinero o cheque electrónico):

```
<?xml versión='1.0'?>
<PaymentInfo xmlns='http://ejemplo.org/pagov2'>
  <Nombre>John Smith</Nombre>
  <Límite de tarjeta de crédito='5000' Moneda='USD'>
    <Número>4019 2445 0277 5567</Número>
    <Emisor>Banco de ejemplo</Emisor>
    <Vencimiento>04/02</Vencimiento>
  </Tarjeta de crédito>
</PagoInfo>
```

Este margen representa que John Smith está usando su tarjeta de crédito con un límite de $5,000USD.

2.1.1 Cifrar un elemento XML

¡El número de tarjeta de crédito de Smith es información confidencial! Si la aplicación desea mantener esa información confidencial, puede cifrar el `CreditCard`elemento:

```
<?xml versión='1.0'?>
<PaymentInfo xmlns='http://ejemplo.org/pagov2'>
  <Nombre>John Smith</Nombre>
  <Tipo de datos cifrados='http://www.w3.org/2001/04/xmlenc#Element'
   xmlns='http://www.w3.org/2001/04/xmlenc#'>
    <Datos cifrados>
      <CipherValue>A23B45C56</CipherValue>
    </CipherData>
  </EncryptedData>
</PagoInfo>
```

Al cifrar todo el `CreditCard`elemento desde sus etiquetas de inicio a fin, se oculta la identidad del elemento en sí. (Un espía no sabe si utilizó una tarjeta de crédito o una transferencia de dinero). El `CipherData`elemento contiene la serialización cifrada del `CreditCard`elemento.

2.1.2 Cifrado del contenido del elemento XML (Elementos)

Como escenario alternativo, puede ser útil para los agentes intermediarios saber que John usó una tarjeta de crédito con un límite particular, pero no el número, el emisor y la fecha de vencimiento de la tarjeta. En este caso, el contenido (datos de caracteres o elementos secundarios) del `CreditCard`elemento está cifrado:

```
<?xml versión='1.0'?>
<PaymentInfo xmlns='http://ejemplo.org/pagov2'>
  <Nombre>John Smith</Nombre>
  <Límite de tarjeta de crédito='5000' Moneda='USD'>
    <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
     Tipo ='http://www.w3.org/2001/04/xmlenc#Content'>
      <Datos cifrados>
        <CipherValue>A23B45C56</CipherValue>
      </CipherData>
    </EncryptedData>
  </Tarjeta de crédito>
</PagoInfo>
```

2.1.3 Cifrado del contenido del elemento XML (datos de caracteres)

O considere el escenario en el que toda la información, *excepto* el número de tarjeta de crédito real, puede estar clara, incluido el hecho de que existe el elemento Número:

```
<?xml versión='1.0'?>
<PaymentInfo xmlns='http://ejemplo.org/pagov2'>
  <Nombre>John Smith</Nombre>
  <Límite de tarjeta de crédito='5000' Moneda='USD'>
    <Número>
      <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
       Tipo ='http://www.w3.org/2001/04/xmlenc#Content'>
        <Datos cifrados>
          <CipherValue>A23B45C56</CipherValue>
        </a> CipherDat_
      </a> EncryptedDat_
    </Número>
    <Emisor>Banco de ejemplo</Emisor>
    <Vencimiento>04/02</Vencimiento>
  </Tarjeta de crédito>
</PagoInfo>
```

Ambos `CreditCard`y `Number`están claros, pero el contenido de los datos de caracteres `Number`está cifrado.

2.1.4 Cifrado de datos arbitrarios y documentos XML

Si el escenario de la aplicación requiere que toda la información esté cifrada, todo el documento se cifra como una secuencia de octetos. Esto se aplica a datos arbitrarios, incluidos documentos XML.

```
<?xml versión='1.0'?>
<EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
 MimeType='texto/xml'>
  <Datos cifrados>
    <CipherValue>A23B45C56</CipherValue>
```

```
    </a> CipherDat_
  </a> EncryptedDat_
```

2.1.5 Supercifrado : cifrado de datos cifrados

Un documento XML puede contener cero o más `EncryptedData` elementos. `EncryptedData`no puede ser padre o hijo de otro `EncryptedData`elemento. Sin embargo, los datos reales cifrados pueden ser cualquier cosa, incluidos `EncryptedData`elementos `EncryptedKey` (es decir, supercifrado). Durante el supercifrado de un elemento `EncryptedData`o `EncryptedKey`, se debe cifrar todo el elemento. Cifrar solo el contenido de estos elementos o cifrar elementos secundarios seleccionados es una instancia no válida según el esquema proporcionado. Por ejemplo, considere lo siguiente:

```
  <p ay:PaymentInfoxmlns:pay='http://example.org/pagov2'>
    <Id. de datos cifrados='ED1' xmlns='http://www.w3.org/2001/04/xmlenc#'
     Tipo =' http://www.w3.org/2001/04/xmlenc#Element '>
      <Datos cifrados>
        <CipherValue> originalDatos cifrados</CipherValue>
      </CipherData>
    </EncryptedData>
  </pago:InfoPago>
```

Un supercifrado válido de " `//xenc:EncryptedData[@Id='ED1']`" sería:

```
  <p ay:PaymentInfoxmlns:pay='http://example.org/pagov2'>
    <Id. de datos cifrados='ED2' xmlns='http://www.w3.org/2001/04/xmlenc#'
     Tipo =' http://www.w3.org/2001/04/xmlenc#Element '>
      <Datos cifrados>
        <CipherValue> newDatos cifrados</CipherValue>
      </a> CipherDat_
    </a> EncryptedDat_
  </o> pay:PaymentInf_
```

donde el `CipherValue`contenido de ' `newEncryptedData`' es la codificación base64 de la secuencia de octetos cifrados resultante del cifrado del `EncryptedData`elemento con `Id='ED1'`.

2.2 `EncryptedData`y `EncryptedKey` uso

2.2.1 `EncryptedData`con clave simétrica (`KeyName`)

```
  [s1] <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
        Tipo =' http://www.w3.org/2001/04/xmlenc#Element '/>
  [s2] <Método de cifrado
          Algoritmo='http://www.w3.org/2001/04/xmlenc#tripledes-cbc'/>
  [s3] <ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
  [s4] <ds:KeyName>John Smith</ ds:KeyName>
  [s5] </ds:KeyInfo>
  [s6] <CipherData><CipherValue>DEADBEEF</CipherValue></CipherData>
  [s7] </EncryptedData>
```

`[s1]`El tipo de datos cifrados se puede representar como un valor de atributo para ayudar en el descifrado y el procesamiento posterior. En este caso, los datos cifrados eran un "elemento". Otras alternativas incluyen el "contenido" de un elemento o una secuencia de octetos externa que también puede identificarse mediante los atributos `MimeType`y `Encoding`.

`[s2]`Este (3DES CBC) es un cifrado de clave simétrica.

`[s4]`La clave simétrica tiene un nombre asociado "John Smith".

`[s6] CipherData`contiene un `CipherValue`, que es una secuencia de octetos codificada en base64. Alternativamente, podría contener un `CipherReference`, que es una referencia de URI junto con las transformaciones necesarias para obtener los datos cifrados como una secuencia de octetos.

2.2.2 `EncryptedKey` (`ReferenceList`, `ds:RetrievalMethod`, `CarriedKeyName`)

La siguiente `EncryptedData`estructura es muy similar a la anterior, excepto que esta vez se hace referencia a la clave mediante `ds:RetrievalMethod`:

```
  [t01] <Id. de datos cifrados='ED'
         xmlns='http://www.w3.org/2001/04/xmlenc#'>
  [t02] <Método de cifrado
           Algoritmo='http://www.w3.org/2001/04/xmlenc#aes128-cbc'/>
  [t03] <d s:KeyInfoxmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
  [t04] <ds: RetrievalMethodURI='#EK'
             Escriba="http://www.w3.org/2001/04/xmlenc#EncryptedKey"/>
  [t05] <ds:KeyName>Sally Doe</ds:KeyName>
  [t06] </ds:KeyInfo>
  [t07] <CipherData><CipherValue>DEADBEEF</CipherValue></CipherData>
  [t08] </EncryptedData>
```

`[t02]`Este (AES-128-CBC) es un cifrado de clave simétrica.

`[t04] ds:RetrievalMethod`se utiliza para indicar la ubicación de una clave con tipo `&xenc;EncryptedKey`. La clave (AES) se encuentra en '#EK'.

`[t05] ds:KeyName`proporciona un método alternativo para identificar la clave necesaria para descifrar el archivo `CipherData`. Cualquiera o ambos `ds:KeyName`y `ds:KeyRetrievalMethod` podrían usarse para identificar la misma clave.

Dentro del mismo documento XML, existía una `EncryptedKey` estructura a la que se hacía referencia en `[t04]`:

```
[t09] <Id. de clave cifrada='EK' xmlns='http://www.w3.org/2001/04/xmlenc#'>
[t10] <Método de cifrado
         Algoritmo="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
[t11] <ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
[t12] <ds:KeyName>John Smith</ ds:KeyName>
[t13] </ds:KeyInfo>
[t14] <CipherData><CipherValue>xyzabc</CipherValue></CipherData>
[t15] <Lista de referencias>
[t16] <URI de referencia de datos='#ED'/>
[t17] </ListaReferencia>
[t18] <CarriedKeyName>Sally Doe</CarriedKeyName>
[t19] </EncryptedKey>
```

`[t09]`El `EncryptedKey`elemento es similar al `EncryptedData`elemento excepto que los datos cifrados son siempre un valor clave.

`[t10]`Es `EncryptionMethod`el algoritmo de clave pública RSA.

`[t12]` `ds:KeyName`de "John Smith" es una propiedad de la clave necesaria para descifrar (usando RSA) el archivo `CipherData`.

`[t14]`El `CipherData`'s `CipherValue` es una secuencia de octetos que es procesada (serializada, cifrada y codificada) por un objeto cifrado de referencia `EncryptionMethod`. (Tenga en cuenta que una EncryptedKey `EncryptionMethod`es el algoritmo utilizado para cifrar estos octetos y no habla de qué tipo de octetos son).

`[t15-17]`A `ReferenceList`identifica los objetos cifrados (`DataReference`y `KeyReference`) cifrados con esta clave. Contiene `ReferenceList`una lista de referencias a datos cifrados por la clave simétrica contenida dentro de esta estructura.

`[t18]`El `CarriedKeyName`elemento se utiliza para identificar el valor de la clave cifrada al que puede hacer referencia el `KeyName`elemento en `ds:KeyInfo`. (Dado que los valores de los atributos de ID deben ser exclusivos de un documento, `CarriedKeyName`puede indicar que varias `EncryptedKey`estructuras contienen el mismo valor de clave cifrado para diferentes destinatarios).

3 Sintaxis de cifrado

Esta sección proporciona una descripción detallada de la sintaxis y las funciones del cifrado XML. Las características descritas en esta sección DEBEN implementarse a menos que se indique lo contrario. La sintaxis se define mediante [XML-Schema] con el siguiente preámbulo XML, declaración, entidad interna e importación:

```
Definición del esquema:

<?xml versión="1.0" codificación="utf-8"?>
<!DOCTYPE esquema PUBLIC "-//W3C//DTD XMLSchema 200102//ES"
 "http://www.w3.org/2001/XMLSchema.dtd"
 [
   <!ATTLIST esquema
     xmlns:xenc CDATA #FIXED 'http://www.w3.org/2001/04/xmlenc#'
     xmlns:ds CDATA #FIXED 'http://www.w3.org/2000/09/xmldsig#'>
   <!ENTITY xenc 'http://www.w3.org/2001/04/xmlenc#'>
   <!ENTIDAD % p ''>
   <!ENTIDAD % s ''>
  ]>

<esquema xmlns='http://www.w3.org/2001/XMLSchema' versión='1.0'
        xmlns:ds='http://www.w3.org/2000/09/xmldsig#'
        xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
        targetNamespace='http://www.w3.org/2001/04/xmlenc#'
        elemento FormDefault='calificado'>

  <importar espacio de nombres='http://www.w3.org/2000/09/xmldsig#'
          esquemaLocation='http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-schema.xsd'/>
```

3.1 El `EncryptedType`elemento

`EncryptedType`es el tipo abstracto del que se derivan `EncryptedData`y . `EncryptedKey`Si bien estos dos últimos tipos de elementos son muy similares con respecto a sus modelos de contenido, una distinción sintáctica es útil para el procesamiento. La implementación DEBE generar un esquema laxomente válido [esquema XML] `EncryptedData`o `EncryptedKey`según lo especificado en las declaraciones de esquema posteriores. (Tenga en cuenta que la generación válida del esquema laxo significa que el contenido permitido `xsd:ANY`no necesita ser válido). Las implementaciones DEBEN crear estas estructuras XML (`EncryptedType`elementos y sus descendientes/contenido) en el Formulario de normalización C [NFC , Corrección NFC].

```
Definición del esquema:

<complexType nombre=' EncryptedType'abstracto='verdadero'>
  <secuencia>
    <elemento nombre=' EncryptionMethod' tipo=' xenc:EncryptionMethodType'
             minOcurre='0'/>
    <elemento ref=' ds:KeyInfo' minOccurs='0'/>
    <elemento ref=' xenc:CipherData'/>
    <elemento ref='xenc:EncryptionProperties' minOccurs='0'/>
  </secuencia>
  <nombre del atributo='Id' tipo='ID' uso='opcional'/>
  <nombre del atributo='Tipo' tipo='cualquierURI' uso='opcional'/>
  <nombre del atributo='MimeType' tipo='cadena' uso='opcional'/>
  <nombre del atributo='Codificación' tipo='cualquierURI' uso='opcional'/>
 </tipocomplejo>
```

`EncryptionMethod`es un elemento opcional que describe el algoritmo de cifrado aplicado a los datos cifrados. Si el elemento está ausente, el destinatario debe conocer el algoritmo de cifrado o el descifrado fallará.

`ds:KeyInfo`es un elemento opcional, definido por [XML-DSIG], que transporta información sobre la clave utilizada para cifrar los datos. Las secciones posteriores de esta especificación definen nuevos elementos que pueden aparecer como hijos de `ds:KeyInfo`.

`CipherData`es un elemento obligatorio que contiene el `CipherValue`o `CipherReference`con los datos cifrados.

EncryptionPropertiespuede contener información adicional sobre la generación del EncryptedType(por ejemplo, marca de fecha/hora).

Ides un atributo opcional que proporciona el método estándar de asignar una identificación de cadena al elemento dentro del contexto del documento.

Typees un atributo opcional que identifica el tipo de información sobre la forma de texto sin formato del contenido cifrado. Si bien es opcional, esta especificación la aprovecha para el procesamiento obligatorio descrito en *Reglas de procesamiento: descifrado* (sección 4.2). Si el EncryptedDataelemento contiene datos de Type'elemento' o elemento 'contenido' y reemplaza esos datos en el contexto de un documento XML, se recomienda encarecidamente que Typese proporcione el atributo. Sin esta información, el descifrador no podrá restaurar automáticamente el documento XML a su formato de texto sin cifrar original.

MimeTypees un atributo opcional (consultivo) que describe el tipo de medio de los datos que se han cifrado. El valor de este atributo es una cadena con valores definidos por [MIME]. Por ejemplo, si los datos cifrados son un PNG codificado en base64, la transferencia Encodingse puede especificar como ' http://www.w3.org/2000/09/xmldsig#base64 ' y MimeTypecomo 'imagen/png'. Este atributo es puramente consultivo; no MimeTypese requiere validación de la información y no indica que la aplicación de cifrado deba realizar ningún procesamiento adicional. Tenga en cuenta que esta información puede no ser necesaria si ya está vinculada al identificador en el Typeatributo. Por ejemplo, los tipos de Elemento y Contenido definidos en esta especificación son siempre texto codificado en UTF-8.

3.2 El elemento EncryptionMethod

EncryptionMethod es un elemento opcional que describe el algoritmo de cifrado aplicado a los datos cifrados. Si el elemento está ausente, el destinatario debe conocer el algoritmo de cifrado o el descifrado fallará.

```
Definición del esquema:

<complexType nombre='EncryptionMethodType' mixto='verdadero'>
  <secuencia>
    <elemento nombre='KeySize' minOccurs='0' tipo='xenc:KeySizeType'/>
    <elemento nombre='OAEPparams' minOccurs='0' tipo='base64Binary'/>
    <cualquier espacio de nombres='##other' minOccurs='0' maxOccurs='ilimitado'/>
  </secuencia>
  <nombre del atributo='Algoritmo' tipo='cualquierURI' uso='requerido'/>
</tipocomplejo>
```

Los elementos secundarios permitidos de EncryptionMethodestán determinados por el valor específico del Algorithmatributo URI, y el KeySizeelemento secundario siempre está permitido. Por ejemplo, el algoritmo RSA-OAEP (sección 5.4.2) utiliza los elementos ds:DigestMethody . OAEPparams(Confiamos en la ANYconstrucción del esquema porque no es posible especificar el contenido del elemento en función del valor de un atributo).

La presencia de cualquier elemento secundario EncryptionMethod que no esté permitido por el algoritmo o la presencia de un KeySizeelemento secundario inconsistente con el algoritmo DEBE tratarse como un error. (Todos los URI de algoritmo especificados en este documento implican un tamaño de clave, pero esto no es cierto en general. Los algoritmos de cifrado de flujo más populares utilizan claves de tamaño variable).

3.3 El CipherDataelemento

Es CipherDataun elemento obligatorio que proporciona los datos cifrados. Debe contener la secuencia de octetos cifrados como texto codificado en base64 del CipherValueelemento o proporcionar una referencia a una ubicación externa que contenga la secuencia de octetos cifrados a través del CipherReferenceelemento.

```
Definición del esquema:

<elemento nombre=' CipherData' tipo=' xenc:CipherDataType'/>
<nombre de tipo complejo=' CipherDataType'>
   <elección>
     <elemento nombre=' CipherValue' tipo='base64Binary'/>
     <elemento ref=' xenc:CipherReference'/>
   </elección>
 </tipocomplejo>
```

3.3.1 El CipherReferenceelemento

Si CipherValueno se suministra directamente, CipherReferenceidentifica una fuente que, cuando se procesa, produce la secuencia de octetos cifrada.

El valor real se obtiene de la siguiente manera. Contiene CipherReference URIun identificador al que se le ha desreferenciado. Si el CipherReferenceelemento contiene una secuencia OPCIONAL de Transforms, los datos resultantes de desreferenciar el URI se transforman según lo especificado para producir el valor de cifrado deseado. Por ejemplo, si el valor está codificado en base64 dentro de un documento XML; las transformaciones podrían especificar una expresión XPath seguida de una decodificación base64 para extraer los octetos.

La sintaxis de URIy Transformses similar a la de [XML-DSIG]. Sin embargo, existe una diferencia entre el procesamiento de firma y cifrado. En [XML-DSIG], tanto el procesamiento de generación como el de validación comienzan con los mismos datos de origen y realizan esa transformación en el mismo orden. En el cifrado, el descifrador sólo tiene los datos cifrados y las transformaciones especificadas se enumeran para el descifrador, en el orden necesario para obtener los octetos. En consecuencia, debido a que tiene una semántica diferente, Transformsestá en el &xenc;espacio de nombres.

Por ejemplo, si el valor de cifrado relevante se captura dentro de un CipherValueelemento dentro de un documento XML diferente, CipherReferencepodría tener el siguiente aspecto:

```
<CipherReference URI="http://www.example.com/CipherValues.xml">
  <Transformaciones>
    <ds:Transformar
     Algoritmo="http://www.w3.org/TR/1999/REC-xpath-19991116">
        <ds:XPath xmlns:rep="http://www.example.org/repositorio">
          self::text()[padre::rep:CipherValue[@Id="ejemplo1"]]
        </ds:XPath>
    </ds:Transformar>
```

```
      <ds:Algoritmo de transformación="http://www.w3.org/2000/09/xmldsig#base64"/>
    </Transforma>
  </CipherReference>
```

Las implementaciones DEBEN admitir la `CipherReference`función y la misma codificación URI, desreferenciación, esquema y códigos de respuesta HTTP que los de [XML-DSIG]. La `Transform`característica y los algoritmos de transformación particulares son OPCIONALES.

```
Definición del esquema:

<elemento nombre=' CipherReference' tipo=' xenc:CipherReferenceType'/>
 <nombre de tipo complejo=' CipherReferenceType'>
     <secuencia>
       <elemento nombre='Transforms' tipo='xenc:TransformsType' minOccurs='0'/>
     </secuencia>
     <nombre de atributo='URI' tipo='cualquierURI' uso='requerido'/>
 </tipocomplejo>

   <nombre de tipo complejo='Tipo de transformación'>
     <secuencia>
       <elemento ref='ds:Transform' maxOccurs='ilimitado'/>
     </secuencia>
    </tipocomplejo>
```

3.4 El `EncryptedData`elemento

El `EncryptedData`elemento es el elemento central de la sintaxis. Su `CipherData`elemento secundario no solo contiene los datos cifrados, sino que también es el elemento que reemplaza al elemento cifrado o sirve como raíz del nuevo documento.

```
Definición del esquema:

<elemento nombre=' EncryptedData' tipo=' xenc:EncryptedDataType'/>
<nombre de tipo complejo=' EncryptedDataType'>
  <Contenido complejo>
    <base de extensión=' xenc:EncryptedType'>
    </extensión>
  </complexContent>
</tipocomplejo>
```

3.5 Extensiones al`ds:KeyInfo` elemento

Hay tres formas de `CipherData`proporcionar el material de claves necesario para descifrar:

1. El elemento `EncryptedData`o `EncryptedKey`especifica el material de clave asociado a través de un elemento secundario de `ds:KeyInfo`. Todos los elementos secundarios de ds: `KeyInfo`especificados en [XML-DSIG] PUEDEN usarse como calificados:
   1. La compatibilidad con `ds:KeyValue`es OPCIONAL y puede usarse para transportar claves públicas, como los valores clave Diffie-Hellman (sección 5.5.1). (Obviamente NO SE RECOMIENDA incluir la clave de descifrado de texto plano, ya sea una clave privada o secreta).
   2. Se RECOMIENDA el soporte de `ds:KeyName`para hacer referencia a un `.EncryptedKey CarriedKeyName`
   3. `ds:RetrievalMethod`Se REQUIERE soporte para el mismo documento .

   Además, proporcionamos dos elementos secundarios adicionales: las aplicaciones DEBEN ser compatibles `EncryptedKey` (sección 3.5.1) y PUEDEN ser compatibles `AgreementMethod`(sección 5.5).
2. Un elemento separado (no dentro de `ds:KeyInfo`) `EncryptedKey`puede especificar el `EncryptedData`o `EncryptedKey`al cual se aplicará su clave descifrada a través de `DataReference`o `KeyReference`(sección 3.6).
3. El material de claves lo puede determinar el destinatario según el contexto de la aplicación y, por lo tanto, no es necesario mencionarlo explícitamente en el XML transmitido.

3.5.1 El `EncryptedKey`elemento

Identificador
`Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey"`

(Esto se puede usar dentro de un `ds:RetrievalMethod`elemento para identificar el tipo de referente).

El `EncryptedKey`elemento se utiliza para transportar claves de cifrado desde el origen hasta uno o varios destinatarios conocidos. Puede usarse como un documento XML independiente, colocarse dentro de un documento de aplicación o aparecer dentro de un `EncryptedData`elemento como hijo de un `ds:KeyInfo` elemento. El valor de la clave siempre está cifrado para los destinatarios. Cuando `EncryptedKey`se descifra, los octetos resultantes se ponen a disposición del `EncryptionMethod`algoritmo sin ningún procesamiento adicional.

```
Definición del esquema:

<elemento nombre=' EncryptedKey' tipo=' xenc:EncryptedKeyType'/>
<nombre de tipo complejo=' EncryptedKeyType'>
  <Contenido complejo>
    <base de extensión=' xenc:EncryptedType'>
      <secuencia>
        <elemento ref=' xenc:ReferenceList' minOccurs='0'/>
        <elemento nombre=' CarriedKeyName' tipo='cadena' minOccurs='0'/>
      </secuencia>
      <nombre del atributo='Destinatario' tipo='cadena' uso='opcional'/>
    </extensión>
  </complexContent>
</tipocomplejo>
```

`ReferenceList`es un elemento opcional que contiene punteros a datos y claves cifradas con esta clave. La lista de referencias puede contener múltiples referencias `EncryptedKey`y `EncryptedData`elementos. Esto se hace usando elementos `KeyReference`y `DataReference`respectivamente. Estos se definen a continuación.

CarriedKeyNamees un elemento opcional para asociar un nombre legible por el usuario con el valor clave. Esto luego se puede usar para hacer referencia a la clave usando el ds:KeyNameelemento dentro de ds:KeyInfo. La misma CarriedKeyNameetiqueta, a diferencia de un tipo de identificación, puede aparecer varias veces dentro de un solo documento. El valor de la clave debe ser el mismo en todos EncryptedKeylos elementos identificados con la misma CarriedKeyNameetiqueta dentro de un único documento XML. Tenga en cuenta que debido a que los espacios en blanco son significativos en el valor del ds:KeyName elemento, los espacios en blanco también son significativos en el valor del CarriedKeyNameelemento.

Recipientes un atributo opcional que contiene una pista sobre a qué destinatario está destinado este valor de clave cifrada. Su contenido depende de la aplicación.

El Typeatributo heredado de EncryptedType se puede utilizar para especificar aún más el tipo de clave cifrada si EncryptionMethod Algorithmno define una codificación/representación inequívoca. (Tenga en cuenta que todos los algoritmos de esta especificación tienen una representación inequívoca para sus estructuras clave asociadas).

3.5.2 El ds:RetrievalMethodelemento

El con a de ' ' proporciona una manera de expresar un enlace a un elemento que contiene la clave necesaria para descifrar el elemento asociado con un o . El de este tipo es siempre hijo del elemento y puede aparecer varias veces. Si hay más de una instancia de a de este tipo, entonces los objetos a los que se hace referencia deben contener el mismo valor de clave, posiblemente cifrado de diferentes maneras o para diferentes destinatarios.ds:RetrievalMethod [XML-DSIG]Typehttp://www.w3.org/2001/04/xmlenc#EncryptedKeyEncryptedKeyCipherDataEncryptedDataEncryptedKeyds:RetrievalMethodds:KeyInfo

```
Definición del esquema:

<!--
    <nombre del atributo='Tipo' tipo='cualquierURI' uso='opcional'
    fijo='http://www.w3.org/2001/04/xmlenc# EncryptedKey' />
-->
```

3.6 El ReferenceListelemento

ReferenceListes un elemento que contiene punteros desde un valor clave de an EncryptedKeya elementos cifrados por ese valor clave (EncryptedDatao EncryptedKeyelementos).

```
Definición del esquema:

<nombre del elemento='Lista de referencias'>
  <tipocomplejo>
    <elección minOccurs='1' maxOccurs='ilimitado'>
      <elemento nombre='DataReference' tipo='xenc:ReferenceType'/>
      <elemento nombre='KeyReference' tipo='xenc:ReferenceType'/>
    </elección>
  </tipocomplejo>
</elemento>

<nombre de tipo complejo=' ReferenceType'>
  <secuencia>
    <cualquier espacio de nombres='##other' minOccurs='0' maxOccurs='ilimitado'/>
  </secuencia>
  <nombre de atributo='URI' tipo='cualquierURI' uso='requerido'/>
</tipocomplejo>
```

DataReferenceLos elementos se utilizan para referirse a EncryptedDataelementos que se cifraron utilizando la clave definida en el EncryptedKeyelemento adjunto. Pueden aparecer varios DataReferenceelementos si EncryptedDataexisten varios elementos cifrados con la misma clave.

KeyReferenceLos elementos se utilizan para referirse a EncryptedKeyelementos que se cifraron utilizando la clave definida en el EncryptedKeyelemento adjunto. Pueden aparecer varios KeyReferenceelementos si EncryptedKeyexisten varios elementos cifrados con la misma clave.

Para ambos tipos de referencias, se pueden especificar opcionalmente elementos secundarios para ayudar al destinatario a recuperar los elementos EncryptedKeyy/o EncryptedData. Estos podrían incluir información como transformaciones XPath, transformaciones de descompresión o información sobre cómo recuperar los elementos de una instalación de almacenamiento de documentos. Por ejemplo:

```
<Lista de referencia>
  <URI de referencia de datos="#factura34">
    <ds:Transformaciones>
      <ds:Algoritmo de transformación="http://www.w3.org/TR/1999/REC-xpath-19991116">
        <ds:XPath xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
            self::xenc:EncryptedData[@Id="ejemplo1"]
        </ds:XPath>
      </ds:Transformar>
    </ds:Transformaciones>
  </ReferenciaDeDatos>
</ReferenciaLista>
```

3.7 El EncryptionProperties elemento

Identificador
Type="http://www.w3.org/2001/04/xmlenc#EncryptionProperties"

(Esto se puede usar dentro de un ds:Referenceelemento para identificar el tipo de referente).

Se pueden colocar elementos de información adicional relacionados con la generación de EncryptedDatao en un elemento (por ejemplo, marca de fecha/hora o el número de serie del hardware criptográfico utilizado durante el cifrado). El atributo identifica la estructura que se describe. permite la inclusión de atributos del espacio de nombres XML que se incluirán (es decir, , y).EncryptedKeyEncryptionPropertyTargetEncryptedTypeanyAttributexml:spacexml:langxml:base

```
Definición del esquema:

<elemento nombre='EncryptionProperties' tipo='xenc:EncryptionPropertiesType'/>
<complexType nombre='EncryptionPropertiesType'>
  <secuencia>
    <elemento ref='xenc:EncryptionProperty' maxOccurs='ilimitado'/>
  </secuencia>
  <nombre del atributo='Id' tipo='ID' uso='opcional'/>
</tipocomplejo>

  <elemento nombre='EncryptionProperty' tipo='xenc:EncryptionPropertyType'/>
  <complexType nombre='EncryptionPropertyType' mixto='verdadero'>
    <elección maxOccurs='ilimitado'>
      <cualquier espacio de nombres='##other' ProcessContents='lax'/>
    </elección>
    <nombre del atributo='Destino' tipo='cualquierURI' uso='opcional'/>
    <nombre del atributo='Id' tipo='ID' uso='opcional'/>
    <anyAttribute namespace="http://www.w3.org/XML/1998/namespace"/>
  </tipocomplejo>
```

4 reglas de procesamiento

Esta sección describe las operaciones que se realizarán como parte del procesamiento de cifrado y descifrado mediante implementaciones de esta especificación. Los requisitos de conformidad se especifican en los siguientes roles:

Solicitud
: La aplicación que solicita la implementación de un cifrado XML mediante el suministro de datos y parámetros necesarios para su procesamiento.

cifrador
: Una implementación de cifrado XML con la función de cifrar datos.

Descifrador
: Una implementación de cifrado XML con la función de descifrar datos.

4.1 Cifrado

Para que cada elemento de datos se cifre como `EncryptedData`o `EncryptedKey`(elementos derivados de `EncryptedType`), el **cifrador** debe:

1. Seleccione el algoritmo (y los parámetros) que se utilizarán para cifrar estos datos.
2. Obtener y (opcionalmente) representar la clave.
    1. Si la clave debe identificarse (mediante nombre, URI o incluirse en un elemento secundario), construya la clave `ds:KeyInfo`según corresponda (p. ej. `ds:KeyName`, `ds:KeyValue`, `ds:RetrievalMethod`, etc.)
    2. Si se va a cifrar la clave en sí, construya un `EncryptedKey`elemento aplicando recursivamente este proceso de cifrado. El resultado puede ser hijo de `ds:KeyInfo`o puede existir en otro lugar y puede identificarse en el paso anterior.
3. cifrar los datos
    1. Si los datos son un ' elemento ' [XML , sección 3] o un elemento ' contenido ' [XML , sección 3.1], obtenga los octetos serializando los datos en UTF-8 como se especifica en [XML]. (La aplicación DEBE proporcionar datos XML en [NFC].) La serialización PUEDE ser realizada por el **cifrador** . Si el **cifrador** no se serializa, entonces la **aplicación** DEBE realizar la serialización.
    2. Si los datos son de cualquier otro tipo que aún no sean octetos, la **aplicación** DEBE serializarlos como octetos.
    3. Cifre los octetos utilizando el algoritmo y la clave de los pasos 1 y 2.
    4. A menos que el **descifrador** conozca implícitamente el tipo de datos cifrados, el **cifrado** DEBE proporcionar el tipo para la representación.

       La definición de este tipo como vinculado a un identificador especifica cómo obtener e interpretar los octetos de texto plano después del descifrado. Por ejemplo, el identificador podría indicar que los datos son una instancia de otra aplicación (por ejemplo, alguna aplicación de compresión XML) que debe procesarse más. O, si los datos son una secuencia de octetos simple, PUEDEN describirse con los atributos `MimeType`y `Encoding`. Por ejemplo, los datos pueden ser un documento XML (`MimeType="text/xml"`), una secuencia de caracteres (`MimeType="text/plain"`) o datos de imagen binaria (`MimeType="image/png"`).
4. Construya la estructura `EncryptedType`(`EncryptedData`o `EncryptedKey`):

   Una `EncryptedType`estructura representa toda la información analizada anteriormente, incluido el tipo de datos cifrados, el algoritmo de cifrado, los parámetros, la clave, el tipo de datos cifrados, etc.

    1. Si la secuencia de octetos cifrada obtenida en el paso 3 se va a almacenar en el `CipherData`elemento dentro de `EncryptedType`, entonces la secuencia de octetos cifrada se codifica en base64 y se inserta como contenido de un `CipherValue`elemento.
    2. Si la secuencia de octetos cifrada se va a almacenar externamente a la `EncryptedType`estructura, almacene o devuelva la secuencia de octetos cifrada y represente el URI y las transformaciones (si las hay) necesarias para que el descifrador recupere la secuencia de octetos cifrada dentro de un `CipherReference`elemento.
5. Procesar datos cifrados
    1. Si los `Type`datos cifrados son " elemento " o elemento " contenido ", entonces el **cifrado** DEBE poder devolver el `EncryptedData`elemento a la **aplicación** . La **aplicación** PUEDE utilizar esto como elemento de nivel superior en un nuevo documento XML o insertarlo en otro documento XML, lo que puede requerir una nueva codificación.

       El **cifrado** DEBE poder reemplazar el 'elemento' o 'contenido' no cifrado con el elemento EncryptedData. Cuando una **aplicación** requiere que se reemplace un elemento o contenido XML, proporciona el contexto del documento XML además de identificar el elemento o contenido que se va a reemplazar. El **cifrador** elimina el elemento o contenido identificado y lo inserta `EncryptedData`en su lugar.

       (Nota: si el `Type`"contenido" es el documento resultante del descifrado no estará bien formado si (a) el texto sin formato original no estaba bien formado (por ejemplo, PCDATA por sí solo no está bien formado) y (b) el `EncryptedData`elemento era anteriormente el elemento raíz del documento)
    2. Si el elemento `Type`de los datos cifrados *no es* " elemento " o " contenido " del elemento, entonces el **cifrador** siempre DEBE devolver el `EncryptedData`elemento a la **aplicación** . La **aplicación** PUEDE utilizar esto como elemento de nivel superior en un

nuevo documento XML o insertarlo en otro documento XML, lo que puede requerir una nueva codificación.

4.2 Descifrado

Para que cada `EncryptedType`elemento derivado (es decir, `EncryptedData`o `EncryptedKey`) se descifre, el **descifrador** debe:

1. Procesar el elemento para determinar el algoritmo, parámetros y `ds:KeyInfo`elemento a utilizar. Si se omite alguna información, la **aplicación** DEBE proporcionarla.
2. Localice la clave de cifrado de datos según el `ds:KeyInfo` elemento, que puede contener uno o más elementos secundarios. Estos niños no tienen ningún orden de procesamiento implícito. Si la clave de cifrado de datos está cifrada, busque la clave correspondiente para descifrarla. (Este puede ser un paso recursivo ya que la clave de cifrado de clave puede estar cifrada). O bien, se puede recuperar la clave de cifrado de datos de un almacén local utilizando los atributos proporcionados o el enlace implícito.
3. Descifre los datos contenidos en el `CipherData`elemento.
    1. Si hay un `CipherValue`elemento secundario presente, entonces se recupera el valor de texto asociado y se decodifica en base64 para obtener la secuencia de octetos cifrada.
    2. Si hay un `CipherReference`elemento secundario presente, el URI y las transformaciones (si las hay) se utilizan para recuperar la secuencia de octetos cifrados.
    3. La secuencia de octetos cifrada se descifra utilizando el algoritmo/parámetros y el valor de clave ya determinados en los pasos 1 y 2.
4. Procesar datos descifrados de `Type`' elemento ' o elemento ' contenido '.
    1. La secuencia de octetos de texto sin cifrar obtenida en el paso 3 se interpreta como datos de caracteres codificados en UTF-8.
    2. El **descifrador** DEBE poder devolver el valor `Type`y los datos de caracteres XML codificados en UTF-8. NO ES NECESARIO que el **descifrador** realice la validación en el XML serializado.
    3. El **descifrador** DEBE admitir la capacidad de reemplazar el `EncryptedData`elemento con el ' elemento ' o el elemento ' contenido ' descifrado representado por los caracteres codificados en UTF-8. NO ES NECESARIO que el **descifrador** realice la validación del resultado de esta operación de reemplazo.

       La aplicación proporciona el contexto del documento XML e identifica el `EncryptedData`elemento que se reemplaza. Si el documento en el que se realiza el reemplazo no es UTF-8, el **descifrador** DEBE transcodificar los caracteres codificados en UTF-8 a la codificación de destino.
5. Procesar datos descifrados si `Type`no están especificados o *no* son ' elemento ' o elemento ' contenido '.
    1. La secuencia de octetos de texto sin cifrar obtenida en **el paso 3** DEBE devolverse a la **aplicación** para su posterior procesamiento junto con los valores de atributo, y cuando se `Type`especifiquen `MimeType`. y son asesores. El valor es normativo ya que puede contener información necesaria para el procesamiento o interpretación de los datos por parte de la aplicación.`EncodingMimeTypeEncodingType`
    2. Tenga en cuenta que este paso incluye el procesamiento de datos descifrados de un archivo `EncryptedKey`. La secuencia de octetos de texto sin cifrar representa un valor clave y la aplicación la utiliza para descifrar otros `EncryptedType`elementos.

4.3 Cifrado XML

Las operaciones de cifrado y descifrado se transforman en octetos. La **aplicación** es responsable de ordenar el XML de manera que pueda serializarse en una secuencia de octetos, cifrarse, descifrarse y ser de utilidad para el destinatario.

Por ejemplo, si la aplicación desea canonicalizar sus datos o codificar/comprimir los datos en un formato de empaquetado XML, la aplicación necesita ordenar el XML en consecuencia e identificar el tipo resultante mediante el `EncryptedData Type`atributo. La probabilidad de que el descifrado y el procesamiento posterior sean exitosos dependerán del soporte del destinatario para el tipo dado. Además, si se pretende que los datos se procesen antes del cifrado y después del descifrado (por ejemplo, validación de firma XML [XML-DSIG] o una transformación XSLT), la aplicación de cifrado debe tener cuidado de preservar la información necesaria para el éxito de ese proceso.

Para fines de interoperabilidad, DEBEN implementarse los siguientes tipos de modo que una implementación pueda tomar como entrada y producir como salida datos que coincidan con las reglas de producción 39 y 43 de [XML] :

elemento ' http://www.w3.org/2001/04/xmlenc#Element '
"[39] elemento ::= `EmptyElemTag` | Contenido de STag ETag "
contenido ' http://www.w3.org/2001/04/xmlenc#Content '
"[43] contenido ::= `CharData`? ((elemento | Referencia | CDSect | PI | Comentario) `CharData`?)*"

Las siguientes secciones contienen especificaciones para descifrar, reemplazar y serializar contenido XML (es decir, `Type`' elemento ' o elemento ' contenido ') utilizando el modelo de datos [XPath]. Estas secciones no son normativas y son OPCIONALES para los implementadores de esta especificación, pero pueden ser referenciadas normativamente y OBLIGATORIAS para otras especificaciones que requieren un procesamiento consistente para aplicaciones, como [XML-DSIG-Decrypt] .

4.3.1 Una implementación de Decrypt (no normativa)

Donde *P* es el contexto en el que se debe analizar el XML serializado (un nodo de documento o nodo de elemento) y *O* es la secuencia de octetos que representa caracteres codificados en UTF-8 resultantes del paso 4.3 en el *Procesamiento de descifrado* (sección 4.2). *Y* es un conjunto de nodos que representa el contenido descifrado obtenido mediante los siguientes pasos:

1. Sea *C* el **contexto de análisis** de un hijo de *P* , que consta de los siguientes elementos:
    - Prefijo y nombre del espacio de nombres de cada espacio de nombres que está dentro del alcance de *P* .
    - Nombre y valor de cada entidad general que es efectiva para el documento XML que causa *P* .
2. Envuelva el flujo de octetos descifrado *O* en el contexto *C* como se especifica en *Ajuste de texto* .
3. Analice el flujo de octetos envueltos como se describe en *El modelo de procesamiento de referencia* (sección 4.3.3.2) de [Firma XML], lo que da como resultado un conjunto de nodos.
4. *Y* es el conjunto de nodos obtenido al eliminar el nodo raíz, el nodo del elemento envolvente y su conjunto asociado de nodos de atributos y espacios de nombres del conjunto de nodos obtenido en el Paso 3.

4.3.2 Una implementación de descifrado y reemplazo (no normativa)

Donde *X* es el conjunto de nodos [XPath] correspondiente a un documento XML y *e* es un `EncryptedData`nodo de elemento en *X* .

1. *Z* es un conjunto de nodos [XPath] idéntico a X excepto donde el nodo de elemento *e* es un EncryptedDatatipo de elemento. En ese caso:
    1. Descifre *e* en el contexto de su nodo principal como se especifica en la *Implementación de descifrado* (sección 4.3.1), lo que produce *Y* , un conjunto de nodos [XPath].
    2. Incluya *Y* en lugar de *e* y sus descendientes en *X* . Dado que [XPath] no define métodos para reemplazar conjuntos de nodos de diferentes documentos, el resultado DEBE ser equivalente a reemplazar e con el flujo de octetos resultante de su descifrado en la forma serializada de X y *analizar* el documento. Sin embargo, el método real para realizar esta operación queda en manos del implementador.

4.3.3 Serialización de XML (no normativo)

Consideraciones sobre el espacio de nombres predeterminado

En *Cifrado de XML* (sección 4.1, paso 3.1), al serializar un fragmento XML DEBE tenerse especial cuidado con respecto a los espacios de nombres predeterminados. Si los datos se descifran posteriormente en el contexto de un documento XML principal, la serialización puede producir elementos en el espacio de nombres incorrecto. Considere el siguiente fragmento de XML:

```
<Documento xmlns="http://ejemplo.org/">
  <ToBeEncrypted xmlns="" />
</Documento>
```

La serialización del ToBeEncryptedfragmento de elemento mediante [XML-C14N] daría como resultado los caracteres " <ToBeEncrypted></ToBeEncrypted>" como una secuencia de octetos. El documento cifrado resultante sería:

```
<Documento xmlns="http://ejemplo.org/">
  <EncryptedData xmlns="...">
    <!-- Contiene el cifrado
         "<ToBeEncrypted></ToBeEncrypted>" -->
  </EncryptedData>
</Documento>
```

Descifrar y reemplazar el EncryptedDatacontenido de este documento produciría el siguiente resultado incorrecto:

```
<Documento xmlns="http://ejemplo.org/">
  <Para ser cifrado/>
</Documento>
```

Este problema surge porque la mayoría de las serializaciones XML asumen que los datos serializados se analizarán directamente en un contexto donde no existe una declaración de espacio de nombres predeterminada. En consecuencia, no declaran de forma redundante el espacio de nombres predeterminado vacío con una extensión xmlns="". Sin embargo, si los datos serializados se analizan en un contexto donde una declaración de espacio de nombres predeterminada está dentro del alcance (por ejemplo, el contexto de análisis de una *implementación de A Decrypt* (sección 4.3.1)), entonces puede afectar la interpretación de los datos serializados.

Para resolver este problema, se PUEDE aumentar un algoritmo de canonicalización de la siguiente manera para usarlo como serializador de cifrado XML:

- xmlns=""Se DEBE emitir una declaración de espacio de nombres predeterminada con un valor vacío (es decir,) donde normalmente el algoritmo de canonicalización la suprimiría.

Si bien es posible que el resultado no esté en la forma canónica adecuada, esto es inofensivo ya que el flujo de octetos resultante no se utilizará directamente en un cálculo del valor de firma [Firma XML]. Volviendo al ejemplo anterior con nuestro nuevo aumento, el ToBeEncryptedelemento se serializaría de la siguiente manera:

```
<ToBeEncrypted xmlns=""></ToBeEncrypted>
```

Cuando se procesa en el contexto del documento principal, este fragmento serializado se analizará e interpretará correctamente.

Este aumento se puede aplicar retroactivamente a una implementación de canonicalización existente canonicalizando cada nodo vértice y sus descendientes del conjunto de nodos, insertándolos xmlns=""en los puntos apropiados y concatenando los flujos de octetos resultantes.

Consideraciones sobre atributos XML

Se debe prestar una atención similar entre la relación de un fragmento y el contexto en el que se inserta a los atributos xml:base, xml:langy xml:spacecomo se menciona en las *Consideraciones de seguridad* de [XML-exc-C14N]. Por ejemplo, si el elemento:

```
<Bongo href="ejemplo.xml"/>
```

se toma de un contexto y se serializa sin atributo xml:base[XML-Base] y se analiza en el contexto del elemento:

```
<Baz xml:base="http://ejemplo.org/"/>
```

el resultado será:

```
<Baz xml:base="http://example.org/"><Bongo href="ejemplo.xml"/></Baz>
```

Bongo's hrefse interpreta posteriormente como " http://example.org/example.xml". Si este no es el URI correcto, Bongodebería haberse serializado con su propio xml:baseatributo.

Desafortunadamente, la recomendación de que se emita un valor vacío para separar el espacio de nombres predeterminado del fragmento del contexto en el que se inserta no se puede realizar para los atributos xml:basey xml:space. (*El error 41* de la errata de especificación XML 1.0 de segunda edición aclara que un valor de cadena vacío del atributo xml:lang *se* considera como si "no hubiera información de idioma disponible, como si xml:langno se hubiera especificado".) La interpretación de un valor vacío para los atributos xml:baseo xml:spaceno

están definidos o mantienen el valor contextual. En consecuencia, las aplicaciones DEBEN garantizar (1) que los fragmentos que se van a cifrar no dependan de atributos XML, o (2) si son dependientes y se pretende que el documento resultante sea válido [XML], la definición del fragmento *permite* la presencia del atributo y que los atributos tengan valores no vacíos.

4.3.4 Ajuste de texto (no normativo)

Esta sección especifica el proceso para ajustar texto en un contexto de análisis determinado. El proceso se basa en la propuesta de Richard Tobin [Tobin] para construir el conjunto de información [XML-Infoset] de una entidad externa.

El proceso consta de los siguientes pasos:

1. Si el contexto de análisis contiene entidades generales, emita una declaración de tipo de documento que proporcione declaraciones de entidades.
2. Emita una dummyetiqueta de inicio de elemento con atributos de declaración de espacio de nombres que declaren todos los espacios de nombres en el contexto de análisis.
3. Emitir el texto.
4. Emitir una dummyetiqueta final de elemento.

En los pasos anteriores, la declaración del tipo de documento y dummy las etiquetas de elementos DEBEN codificarse en UTF-8.

Considere el siguiente documento que contiene un EncryptedData elemento:

```
<!DOCTYPE Documento [
  <!ENTITY dsig "http://www.w3.org/2000/09/xmldsig#">
]>
<Documento xmlns="http://ejemplo.org/">
  <foo:Cuerpo xmlns:foo="http://example.org/foo">
    <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
                  Escriba=" http://www.w3.org/2001/04/xmlenc#Element ">
      ...
    </EncryptedData>
  </foo:Cuerpo>
</Documento>
```

Si el EncryptedDataelemento se alimenta y se descifra en el texto " <One><foo:Two/></One>", entonces el formato ajustado es el siguiente:

```
<!DOCTYPE ficticio [
  <!ENTITY dsig "http://www.w3.org/2000/09/xmldsig#">
]>
<dummy xmlns="http://ejemplo.org/"
       xmlns:foo="http://example.org/foo"><Uno><foo:Two/></Uno></dummy>
```

5. Algoritmos

Esta sección analiza los algoritmos utilizados con la especificación de cifrado XML. Las entradas contienen el identificador que se utilizará como valor del Algorithmatributo del EncryptionMethodelemento u otro elemento que represente el rol del algoritmo, una referencia a la especificación formal, definiciones para la representación de claves y los resultados de las operaciones criptográficas cuando corresponda, y información general. comentarios de aplicabilidad.

5.1 Identificadores de algoritmos y requisitos de implementación

Todos los algoritmos enumerados a continuación tienen parámetros implícitos según su función. Por ejemplo, los datos que se van a cifrar o descifrar, el material de claves y la dirección de operación (cifrado o descifrado) de los algoritmos de cifrado. Cualquier parámetro adicional explícito de un algoritmo aparece como elementos de contenido dentro del elemento. Dichos elementos secundarios de parámetros tienen nombres de elementos descriptivos, que frecuentemente son específicos del algoritmo, y DEBEN estar en el mismo espacio de nombres que esta especificación de cifrado XML, la especificación de firma XML o en un espacio de nombres específico del algoritmo. Un ejemplo de un parámetro tan explícito podría ser un nonce (cantidad única) proporcionado a un algoritmo de acuerdo clave.

Esta especificación define un conjunto de algoritmos, sus URI y requisitos de implementación. Los niveles de requisitos especificados, como "REQUERIDO" u "OPCIONAL", se refieren a la implementación, no al uso. Además, el mecanismo es extensible y se pueden utilizar algoritmos alternativos.

Tabla de algoritmos

La siguiente tabla enumera las categorías de algoritmos. Dentro de cada categoría, se proporciona un nombre breve, el nivel de requisito de implementación y un URI de identificación para cada algoritmo.

Cifrado de bloques

1. TRIPLEDES REQUERIDOS
   http://www.w3.org/2001/04/xmlenc#tripledes-cbc
2. AES-128 REQUERIDO
   http://www.w3.org/2001/04/xmlenc#aes128-cbc
3. AES-256 REQUERIDO
   http://www.w3.org/2001/04/xmlenc#aes256-cbc
4. OPCIONAL AES-192
   http://www.w3.org/2001/04/xmlenc#aes192-cbc

Cifrado de flujo

1. none
   A continuación se proporcionan sintaxis y recomendaciones para admitir algoritmos especificados por el usuario.

Transporte clave

1. REQUERIDO RSA-v1.5
   http://www.w3.org/2001/04/xmlenc#rsa-1_5
2. REQUERIDO RSA-OAEP
   http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p

Acuerdo clave

1. OPCIONAL Diffie-Hellman
   http://www.w3.org/2001/04/xmlenc#dh

Envoltura de clave simétrica

1. TRIPLEDES REQUERIDOS KeyWrap
   http://www.w3.org/2001/04/xmlenc#kw-tripledes
2. REQUERIDO AES-128 KeyWrap
   http://www.w3.org/2001/04/xmlenc#kw-aes128
3. REQUERIDO AES-256 KeyWrap
   http://www.w3.org/2001/04/xmlenc#kw-aes256
4. OPCIONAL AES-192 KeyWrap
   http://www.w3.org/2001/04/xmlenc#kw-aes192

Resumen del mensaje

1. SHA1 REQUERIDO
   http://www.w3.org/2000/09/xmldsig#sha1
2. RECOMENDADO SHA256
   http://www.w3.org/2001/04/xmlenc#sha256
3. SHA512 OPCIONAL
   http://www.w3.org/2001/04/xmlenc#sha512
4. RIPEMD-160 OPCIONAL
   http://www.w3.org/2001/04/xmlenc#ripemd160

Autenticación de mensajes

1. Firma digital XML RECOMENDADA
   http://www.w3.org/2000/09/xmldsig#

Canonicalización

1. XML canónico OPCIONAL (omite comentarios)
   http://www.w3.org/TR/2001/REC-xml-c14n-20010315
2. XML canónico OPCIONAL con comentarios
   http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments
3. OPCIONAL Canonicalización XML exclusiva (omite comentarios)
   http://www.w3.org/2001/10/xml-exc-c14n#
4. OPCIONAL Canonicalización XML exclusiva con comentarios
   http://www.w3.org/2001/10/xml-exc-c14n#WithComments

Codificación

1. REQUERIDO base64
   http://www.w3.org/2000/09/xmldsig#base64

5.2 Algoritmos de cifrado de bloques

Los algoritmos de cifrado de bloques están diseñados para cifrar y descifrar datos en bloques de varios octetos de tamaño fijo. Sus identificadores aparecen como el valor de los `Algorithm`atributos de `EncryptionMethod`los elementos hijos de `EncryptedData`.

Los algoritmos de cifrado de bloques toman, como argumentos implícitos, los datos que se van a cifrar o descifrar, el material de clave y su dirección de operación. Para todos estos algoritmos especificados a continuación, se requiere un vector de inicialización (IV) codificado con el texto cifrado. Para los algoritmos de cifrado de bloques especificados por el usuario, el IV, si lo hubiera, podría especificarse junto con los datos cifrados, como un elemento de contenido del algoritmo o en cualquier otro lugar.

El IV está codificado con y antes del texto cifrado para los algoritmos siguientes para facilitar la disponibilidad del código de descifrado y enfatizar su asociación con el texto cifrado. Las buenas prácticas criptográficas requieren que se utilice un IV diferente para cada cifrado.

Relleno

Dado que los datos que se cifran son un número arbitrario de octetos, es posible que no sean un múltiplo del tamaño del bloque. Esto se resuelve rellenando el texto sin formato hasta el tamaño del bloque antes del cifrado y deshaciendo el relleno después del descifrado. El algoritmo de relleno consiste en calcular el número de octetos más pequeño distinto de cero, por ejemplo `N`, que debe añadirse al texto sin formato para que sea un múltiplo del tamaño del bloque. Supondremos que el tamaño del bloque es `B` de octetos, por lo que `N`está en el rango de 1 a `B`. Rellene añadiendo al texto sin formato un sufijo de `N-1`bytes de relleno arbitrarios y un byte final cuyo valor sea `N`. Al descifrar, simplemente tome el último byte y, después de verificarlo, elimine esa cantidad de bytes del final del texto cifrado descifrado.

Por ejemplo, supongamos un tamaño de bloque de 8 bytes y un texto sin formato de `0x616263`. El texto sin formato acolchado estaría entonces `0x616263??????????05`donde "??" Los bytes pueden tener cualquier valor. De manera similar, el texto sin formato de `0x2122232425262728`se rellenaría con `0x2122232425262728??????????????08`.

5.2.1 Triple DES

Identificador:
http://www.w3.org/2001/04/xmlenc#tripledes-cbc (REQUERIDO)

ANSI X9.52 [TRIPLEDES] especifica tres operaciones FIPS 46-3 [DES] secuenciales. El cifrado XML TRIPLEDES consta de un cifrado DES, un descifrado DES y un cifrado DES utilizado en el modo Cipher Block Chaining (CBC) con 192 bits de clave y un vector de inicialización (IV) de 64 bits. De los bits clave, los primeros 64 bits se utilizan en la primera operación DES, los segundos 64 bits en la operación DES intermedia y los terceros 64 bits en la última operación DES.

Nota: Cada uno de estos 64 bits de clave contiene 56 bits efectivos y 8 bits de paridad. Por tanto, sólo hay 168 bits operativos de los 192 que se transportan para una clave TRIPLEDES. (Dependiendo del criterio utilizado para el análisis, se puede pensar que la fuerza efectiva de la clave es de 112 bits (debido a los ataques intermedios) o incluso menos).

El texto cifrado resultante tiene el prefijo IV. Si se incluye en la salida XML, está codificado en base64. Un ejemplo de método de cifrado TRIPLEDES es el siguiente:

```
<Método de cifrado
 Algoritmo="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
```

5.2.2 AES

Identificador:
http://www.w3.org/2001/04/xmlenc#aes128-cbc (REQUERIDO)
http://www.w3.org/2001/04/xmlenc#aes192-cbc (OPCIONAL)
http://www.w3.org/2001/04/xmlenc#aes256-cbc (REQUERIDO)

[AES] se utiliza en el modo Cipher Block Chaining (CBC) con un vector de inicialización (IV) de 128 bits. El texto cifrado resultante tiene el prefijo IV. Si se incluye en la salida XML, está codificado en base64. Un ejemplo de método de cifrado AES es el siguiente:

```
<Método de cifrado
 Algoritmo="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
```

5.3 Algoritmos de cifrado de flujo

Los algoritmos de cifrado de flujo simple generan, en función de la clave, un flujo de bytes a los que se aplica XOR con los bytes de datos de texto sin formato para producir el texto cifrado en el cifrado y con los bytes de texto cifrado para producir texto sin formato al descifrar. Normalmente se utilizan para el cifrado de datos y se especifican por el valor del `Algorithm`atributo del `EncryptionMethod`hijo de un `EncryptedData` elemento.

NOTA: Es fundamental que cada clave de cifrado de flujo simple (o clave y vector de inicialización (IV) si también se usa un IV) se use solo una vez. Si alguna vez se usa la misma clave (o clave y IV) en dos mensajes, al aplicar XOR en los dos textos cifrados, puede obtener el XOR de los dos textos sin formato. Esto suele ser muy comprometedor.

En este documento no se especifican algoritmos de cifrado de flujo específicos, pero esta sección se incluye para proporcionar pautas generales.

Los algoritmos de transmisión suelen utilizar el `KeySize`parámetro explícito opcional. En los casos en los que el tamaño de la clave no sea evidente en el URI del algoritmo o en el origen de la clave, como en el uso de métodos de acuerdo de claves, este parámetro establece el tamaño de la clave. Si el tamaño de la clave que se utilizará es evidente y no está de acuerdo con el `KeySize`parámetro, DEBE devolverse un error. La implementación de cualquier algoritmo de flujo es opcional. El esquema para el parámetro KeySize es el siguiente:

```
Definición del esquema:

<nombre de tipo simple='Tipo de tamaño de clave'>
  <restricción base="entero"/>
</tipo simple>
```

5.4 Transporte clave

Los algoritmos de transporte de claves son algoritmos de cifrado de claves públicas especialmente especificados para cifrar y descifrar claves. Sus identificadores aparecen como `Algorithm`atributos de `EncryptionMethod`elementos hijos de `EncryptedKey`. `EncryptedKey`es a su vez hijo de un `ds:KeyInfo`elemento. El tipo de clave que se transporta, es decir el algoritmo en el que se prevé utilizar la clave transportada, viene dado por el `Algorithm`atributo del `EncryptionMethod`hijo `EncryptedData`o `EncryptedKey`padre de este `ds:KeyInfo`elemento.

(Los algoritmos de transporte de claves se pueden utilizar opcionalmente para cifrar datos, en cuyo caso aparecen directamente como el `Algorithm`atributo de un elemento `EncryptionMethod`secundario `EncryptedData`. Debido a que utilizan algoritmos de clave pública directamente, los algoritmos de transporte de claves no son eficientes para el transporte de ninguna cantidad de datos significativamente. más grandes que las claves simétricas.)

El algoritmo de transporte de claves RSA v1.5 que se proporciona a continuación son los que se utilizan junto con TRIPLEDES y la sintaxis de mensajes criptográficos (CMS) de S/MIME [algoritmos CMS]. El algoritmo de transporte de claves RSA v2 que se proporciona a continuación es el que se utiliza junto con AES y CMS [AES-WRAP].

5.4.1 RSA Versión 1.5

Identificador:
http://www.w3.org/2001/04/xmlenc#rsa-1_5 (REQUERIDO)

El algoritmo RSAES-PKCS1-v1_5, especificado en RFC 2437 [PKCS1], no toma parámetros explícitos. Un ejemplo de un `EncryptionMethod`elemento RSA versión 1.5 es:

```
<Método de cifrado
 Algoritmo="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
```

La `CipherValue`clave para dicha clave cifrada es la codificación base64 [MIME] de la cadena de octetos calculada según RFC 2437 [PKCS1 , sección 7.2.1: Operación de cifrado]. Como se especifica en la función EME-PKCS1-v1_5 RFC 2437 [PKCS1 , sección 9.1.2.1], el valor ingresado a la función de transporte de claves es el siguiente:

```
CRIPTAR (PAD (TECLA))
```

donde el relleno tiene la siguiente forma especial:

```
02 | PD* | 00 | llave
```

donde "|" es concatenación, "02" y "00" son octetos fijos del valor hexadecimal correspondiente, PS es una cadena de octetos pseudoaleatorios fuertes [ALEATORIO] de al menos ocho octetos de longitud, que no contienen octetos cero y lo suficientemente larga como para que el valor de la cantidad que se CRIPTA es un octeto más corta que el módulo RSA y "clave" es la clave que se transporta. La clave es de 192 bits para TRIPLEDES y de 128, 192 o 256 bits para AES. La compatibilidad con este algoritmo de transporte de claves para transportar claves de 192 bits es OBLIGATORIA de implementar. El soporte de este algoritmo para transportar otras claves es OPCIONAL. Se RECOMIENDA RSA-OAEP para el transporte de claves AES.

La cadena base64 [MIME] resultante es el valor del nodo de texto secundario del `CipherData`elemento, por ejemplo

```
<Datos de cifrado> IWijxQjUrcXBYoCei4QxjWo9Kg8D3p9tlWoT4
   t0/gyTE96639In0FZFY2/rvP+/bMJ01EArmKZsR5VW3rwoPxw=
</CipherData>
```

5.4.2 RSA-OAEP

Identificador:
http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p (REQUERIDO)

El algoritmo RSAES-OAEP-ENCRYPT, como se especifica en RFC 2437 [PKCS1], toma tres parámetros. Los dos parámetros especificados por el usuario son una función de resumen de mensajes OBLIGATORIA y una cadena de octetos de codificación OPCIONAL `OAEPparams`. La función de resumen de mensajes se indica mediante el `Algorithm`atributo de un `ds:DigestMethod`elemento secundario y la función de generación de máscara, el tercer parámetro, es siempre MGF1 con SHA1 (mgf1SHA1Identifier). Tanto las funciones de resumen de mensajes como de generación de máscaras se utilizan en la operación EME-OAEP-ENCODE como parte de RSAES-OAEP-ENCRYPT. La cadena de octeto de codificación es la decodificación base64 del contenido de un `OAEPparams`elemento secundario opcional. Si no `OAEPparams`se proporciona ningún hijo, se utiliza una cadena nula.

```
Definición del esquema:
     <!-- use estos tipos de elementos como hijos de EncryptionMethod
          cuando se usa con RSA-OAEP -->
     <elemento nombre='OAEPparams' minOccurs='0' tipo='base64Binary'/>
     <elemento ref='ds:DigestMethod' minOccurs='0'/>
```

Un ejemplo de un elemento RSA-OAEP es:

```
<Método de cifrado
   Algoritmo="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
  <OAEPparams> 9lWu3Q== </OAEPparams>
  <ds:Método de resumen
      Algoritmo="http://www.w3.org/2000/09/xmldsig#sha1"/>
<Método de cifrado>
```

La `CipherValue`clave cifrada RSA-OAEP es la codificación base64 [MIME] de la cadena de octetos calculada según RFC 2437 [PKCS1 , sección 7.1.1: Operación de cifrado]. Como se describe en la función EME-OAEP-ENCODE RFC 2437 [PKCS1 , sección 9.1.1.1], el valor ingresado a la función de transporte de claves se calcula usando la función de resumen de mensajes y la cadena especificada en los elementos `DigestMethod`y `OAEPparams` y usando la función de generación de máscaras MGF1. (con SHA1) especificado en RFC 2437. La longitud de salida deseada para EME-OAEP-ENCODE es un byte más corta que el módulo RSA.

El tamaño de la clave transportada es de 192 bits para TRIPLEDES y de 128, 192 o 256 bits para AES. Las implementaciones DEBEN implementar RSA-OAEP para el transporte de claves de 128 y 256 bits. PUEDEN implementar RSA-OAEP para el transporte de otras claves.

5.5 Acuerdo clave

Un algoritmo de acuerdo de clave proporciona la derivación de una clave secreta compartida basada en un secreto compartido calculado a partir de ciertos tipos de claves públicas compatibles tanto del remitente como del destinatario. La información del creador para determinar el secreto se indica mediante un `OriginatorKeyInfo`parámetro opcional secundario de un `AgreementMethod`elemento, mientras que el asociado con el destinatario se indica mediante un parámetro opcional `RecipientKeyInfo`. Una clave compartida se deriva de este secreto compartido mediante un método determinado por el algoritmo de Acuerdo de Clave.

Nota: XML Encryption no proporciona un protocolo de negociación de acuerdos de claves en línea. `AgreementMethod`El creador puede utilizar el elemento para identificar las claves y el procedimiento computacional que se utilizaron para obtener una clave de cifrado compartida. El método utilizado para obtener o seleccionar las claves o algoritmo utilizado para el cálculo del acuerdo está fuera del alcance de esta especificación.

El `AgreementMethod`elemento aparece como contenido de a `ds:KeyInfo`ya que, al igual que otros `ds:KeyInfo`elementos secundarios, produce una clave. Éste, `ds:KeyInfo`a su vez, es hijo de un elemento `EncryptedData`o `EncryptedKey`. El `Algorithm`atributo y `KeySize`el elemento secundario del `EncryptionMethod`elemento bajo este `EncryptedData` o `EncryptedKey`elemento son parámetros implícitos para el cálculo del acuerdo clave. En los casos en que este `EncryptionMethod` algoritmo URI sea insuficiente para determinar la longitud de la clave, `KeySize`DEBE haberse incluido. Además, el remitente puede colocar un `KA-Nonce`elemento debajo `AgreementMethod`para garantizar que se genere material de claves diferente incluso para acuerdos repetidos que utilicen las mismas claves públicas de remitente y destinatario. Por ejemplo:

```
<Datos cifrados>
   <Algoritmo de método de cifrado="Ejemplo:Bloque/Alg"
     <Tamaño de clave>80</Tamaño de clave>
   </Método de cifrado>
   <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
     <Algoritmo Método de Acuerdo="ejemplo:Acuerdo/Algoritmo">
       <KA-Nonce>Zm9v</KA-Nonce>
       <ds:Método de resumen
```

```
      Algorithm="http://www.w3.org/2001/04/xmlenc#sha1"/>
     <OriginatorKeyInfo>
        <ds:KeyValue>....</ds:KeyValue>
      </OriginatorKeyInfo>
      <RecipientKeyInfo>
        <ds:KeyValue>....</ds:KeyValue>
      </RecipientKeyInfo>
    </AgreementMethod>
  </ds:KeyInfo>
  <CipherData>...</CipherData>
</EncryptedData>
```

If the agreed key is being used to wrap a key, rather than data as above, then `AgreementMethod` would appear inside a `ds:KeyInfo` inside an `EncryptedKey` element.

The Schema for `AgreementMethod` is as follows:

```
Schema Definition:

<element name="AgreementMethod" type="xenc:AgreementMethodType"/>
<complexType name="AgreementMethodType" mixed="true">
  <sequence>
    <element name="KA-Nonce" minOccurs="0" type="base64Binary"/>
    <!-- <element ref="ds:DigestMethod" minOccurs="0"/> -->
    <any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
    <element name="OriginatorKeyInfo" minOccurs="0"
             type="ds:KeyInfoType"/>
    <element name="RecipientKeyInfo" minOccurs="0"
             type="ds:KeyInfoType"/>
  </sequence>
  <attribute name="Algorithm" type="anyURI" use="required"/>
</complexType>
```

5.5.1 Valores clave de Diffie-Hellman

Identificador:
http://www.w3.org/2001/04/xmlenc#DHKeyValue (OPCIONAL)

Las claves Diffie-Hellman pueden aparecer directamente dentro de `KeyValue` los elementos u obtenerse mediante `ds:RetrievalMethod`recuperaciones, además de aparecer en certificados y similares. El identificador anterior se puede utilizar como el valor del `Type`atributo de `Reference`o `ds:RetrievalMethod`elementos.

Como se especifica en [ESDH], una clave pública DH consta de hasta seis cantidades, dos números primos grandes p y q, un "generador" g, la clave pública y los parámetros de validación "seed" y "pgenCounter". Estos se relacionan de la siguiente manera: La clave pública = (g**x mod p) donde x es la clave privada correspondiente; p = j*q + 1 donde j >= 2. "seed" y "pgenCounter" son opcionales y se pueden utilizar para determinar si la clave Diffie-Hellman se ha generado de conformidad con el algoritmo especificado en [ESDH] . Debido a que los números primos y el generador se pueden compartir de forma segura entre muchas claves DH, es posible que se conozcan desde el entorno de la aplicación y son opcionales. El esquema para a `DHKeyValue`es el siguiente:

```
 Schema:

<element name="DHKeyValue" type="xenc:DHKeyValueType"/>
<complexType name="DHKeyValueType">
   <sequence>
      <sequence minOccurs="0">
         <element name="P" type="ds:CryptoBinary"/>
         <element name="Q" type="ds:CryptoBinary"/>
         <element name="Generator"type="ds:CryptoBinary"/>
      </sequence>
      <element name="Public" type="ds:CryptoBinary"/>
      <sequence minOccurs="0">
         <element name="seed" type="ds:CryptoBinary"/>
         <element name="pgenCounter" type="ds:CryptoBinary"/>
      </sequence>
   </sequence>
</complexType>
```

5.5.2 Acuerdo clave Diffie-Hellman

Identificador:
http://www.w3.org/2001/04/xmlenc#dh (OPCIONAL)

El protocolo de acuerdo de claves Diffie-Hellman (DH) [ESDH] implica la derivación de información secreta compartida basada en claves DH compatibles del remitente y el destinatario. Dos claves públicas DH son compatibles si tienen el mismo número principal y generador. Si, para el segundo, `Y = g**y mod p`entonces las dos partes pueden calcular el secreto compartido `ZZ = ( g**(x*y) mod p )`aunque cada una conozca sólo su propia clave privada y la clave pública de la otra parte. Los bytes cero iniciales DEBEN mantenerse para `ZZ`que tengan la misma longitud, en bytes, que `p`. El tamaño `p`DEBE ser de al menos 512 bits y `g`al menos 160 bits. Existen muchas otras consideraciones de seguridad complejas en la selección de `g`, `p`y aleatorio `x`como se describe en [ESDH].

El acuerdo clave Diffie-Hellman es opcional de implementar. Un ejemplo de un `AgreementMethod`elemento DH es el siguiente:

```
<Método de acuerdo
    Algoritmo="http://www.w3.org/2001/04/xmlenc#dh"
    ds:xmlns="http://www.w3.org/2000/09/xmldsig#">
  <KA-Nonce>Zm9v</KA-Nonce>
  <ds:Método de resumen
   Algoritmo="http://www.w3.org/2000/09/xmldsig#sha1"/>
  <Información de clave del originador>
    <ds:X509Data><ds:X509Certificado>
      ...
    </ds:X509Certificate></ds:X509Data>
  </OriginatorKeyInfo>
```

```
  <RecipientKeyInfo><ds:KeyValue>
    ...
  </ds:KeyValue></RecipientKeyInfo>
</Método de acuerdo>
```

Supongamos que el secreto compartido de Diffie-Hellman es la secuencia de octetos `ZZ`. El material de claves compartido necesario se calculará de la siguiente manera:

```
Material de codificación = KM(1) | KM(2) | ...
```

donde "|" es la concatenación de flujo de bytes y

```
KM(contador) = DigestAlg ( ZZ | contador | EncryptionAlg |
              KA-Nonce | Tamaño de clave)
```

`DigestAlg`
: El algoritmo de resumen de mensajes especificado por el `DigestMethod`hijo de `AgreementMethod`.

`EncryptionAlg`
: El URI del algoritmo de cifrado, incluidos los posibles algoritmos de ajuste de claves, en los que se utilizará el material de claves derivado ("Ejemplo: Bloque/Alg" en el ejemplo anterior), no el URI del algoritmo de acuerdo. Este es el valor del `Algorithm` atributo del `EncryptionMethod`hijo de `EncryptedData`o `EncryptedKey`abuelo de `AgreementMethod`.

`KA-Nonce`
: Base64 decodifica el contenido del `KA-Nonce`hijo de `AgreementMethod`, si está presente. Si el `KA-Nonce` elemento está ausente, es nulo.

`Counter`
: Un contador de un byte que comienza en uno y se incrementa en uno. Se expresa como dos dígitos hexadecimales donde las letras de la A a la F están en mayúsculas.

`KeySize`
: El tamaño en bits de la clave que se derivará del secreto compartido como la cadena UTF-8 para el entero decimal correspondiente con solo dígitos en la cadena y sin ceros a la izquierda. Para algunos algoritmos, el tamaño de la clave es inherente al URI. Para otros, como la mayoría de los cifrados de flujo, se debe proporcionar explícitamente.

`(KM(1))`Por ejemplo, el cálculo inicial para el ejemplo `EncryptionMethod`de Acuerdo de clave (sección 5.5) sería el siguiente, donde el valor del contador binario de un byte de 1 está representado por la secuencia UTF-8 de dos caracteres `01`, `ZZ`es el secreto compartido y " `foo`" es la decodificación base64 de " `Zm9v`".

```
SHA-1 (ZZ01Ejemplo:Bloque/Algfoo80)
```

Suponiendo que así `ZZ`sea `0xDEADBEEF`, eso sería

```
SHA-1 (0xDEADBEEF30314578616D706C653A426C6F636B2F416C67666F6F3830)
```

cuyo valor es

```
0x534C9B8C4ABDCB50038B42015A181711068B08C1
```

Cada aplicación de `DigestAlg`para valores sucesivos de `Counter`producirá un número adicional de bytes de material de claves. De la cadena concatenada de uno o más `KM`, se toman suficientes bytes iniciales para satisfacer la necesidad de una clave real y el resto se descarta. Por ejemplo, si `DigestAlg`SHA-1 produce 20 octetos de hash, entonces para AES de 128 bits `KM(1)`se tomarían los primeros 16 bytes y se descartarían los 4 bytes restantes. `KM(1)`Para AES de 256 bits, se tomarían todos los sufijos con los primeros 12 bytes de KM(2) y `KM(2)` se descartarían los 8 bytes restantes.

5.6 Ajuste de clave simétrica

Los algoritmos Symmetric Key Wrap son algoritmos de cifrado de claves secretas compartidas especialmente especificados para cifrar y descifrar claves simétricas. Sus identificadores aparecen como `Algorithm`valores de atributos de `EncryptionMethod`elementos que son hijos de `EncryptedKey`los cuales son a su vez hijos de `ds:KeyInfo` los cuales son a su vez hijos de `EncryptedData`u otro `EncryptedKey`. El tipo de clave que se encapsula se indica mediante el `Algorithm`atributo de `EncryptionMethod`hijo del padre del `ds:KeyInfo`abuelo del que `EncryptionMethod`especifica el algoritmo de encapsulación de clave simétrica.

5.6.1 Suma de comprobación de clave CMS

Algunos algoritmos de ajuste de claves utilizan una suma de comprobación de claves como se define en CMS [CMS-Wrap]. El algoritmo que proporciona un valor de verificación de integridad para la clave que se empaqueta es:

1. Calcule el hash SHA-1 de 20 octetos en la clave que se está empaquetando.
2. Utilice los primeros 8 octetos de este hash como valor de suma de comprobación.

5.6.2 Envoltura de claves Triple DES de CMS

Identificadores y requisitos:
: http://www.w3.org/2001/04/xmlenc#kw-tripledes (REQUERIDO)

Las implementaciones de cifrado XML DEBEN admitir la envoltura TRIPLEDES de claves de 168 bits y, opcionalmente, pueden admitir la envoltura TRIPLEDES de otras claves.

Un ejemplo de un `EncryptionMethod`elemento Key Wrap TRIPLEDES es el siguiente:

```
<EncryptionMethod
 Algorithm="http://www.w3.org/2001/04/xmlenc#kw-tripledes"/>
```

El siguiente algoritmo envuelve (cifra) una clave (la clave envuelta, WK) bajo una clave de cifrado de clave TRIPLEDES (KEK) adoptada de [Algoritmos CMS]:

1. Representa la clave que se encapsula como una secuencia de octetos. Si se trata de una clave TRIPLEDES, son 24 octetos (192 bits) con un bit de paridad impar como bit inferior de cada octeto.
2. Calcule la suma de verificación de la clave CMS (sección 5.6.1), llame a esto CKS.
3. Sea `WKCKS = WK || CKS`, donde || es la concatenación.
4. Genere 8 octetos aleatorios [RANDOM] y llame a esto IV.
5. Cifre WKCKS en modo CBC utilizando KEK como clave y IV como vector de inicialización. Llame a los resultados TEMP1.
6. Deja `TEMP2 = IV || TEMP1`.
7. Invierta el orden de los octetos `TEMP2`y llame al resultado `TEMP3`.
8. Cifre `TEMP3`en modo CBC utilizando `KEK`y un vector de inicialización de `0x4adda22c79e82105`. El texto cifrado resultante es el resultado deseado. Tiene una longitud de 40 octetos si se encapsula una clave de 168 bits.

El siguiente algoritmo desenvuelve (descifra) una clave adoptada de [Algoritmos CMS]:

1. Compruebe si la longitud del texto cifrado es razonable dado el tipo de clave. Debe tener 40 bytes para una clave de 168 bits y 32, 40 o 48 bytes para una clave de 128, 192 o 256 bits. Si la longitud no es compatible o es inconsistente con el algoritmo para el cual está destinada la clave, se devuelve un error.
2. Descifre el texto cifrado con TRIPLEDES en modo CBC utilizando `KEK`y un vector de inicialización (IV) de `0x4adda22c79e82105`. Llame a la salida `TEMP3`.
3. Invierta el orden de los octetos en TEMP3 y llame al resultado `TEMP2`.
4. Descomponga TEMP2 en IV, los primeros 8 octetos y `TEMP1`los octetos restantes.
5. Descifre TEMP1 usando TRIPLEDES en modo CBC usando `KEK` y el IV que se encuentra en el paso anterior. Llame al resultado `WKCKS`.
6. Descomponer `WKCKS`. `CKS`son los últimos 8 octetos y `WK`, la clave envuelta, son los octetos anteriores al `CKS`.
7. Calcule una suma de verificación de clave CMS (sección 5.6.1) sobre `WK`y compárela con la `CKS` extraída en el paso anterior. Si no son iguales, devuelve error.
8. `WK`es la clave empaquetada, ahora extraída para usarla en el descifrado de datos.

5.6.3 Ajuste de clave AES

Identificadores y requisitos:

http://www.w3.org/2001/04/xmlenc#kw-aes128 (REQUERIDO)
http://www.w3.org/2001/04/xmlenc#kw-aes192 (OPCIONAL)
http://www.w3.org/2001/04/xmlenc#kw-aes256 (REQUERIDO)

La implementación del ajuste de claves AES se describe a continuación, según lo sugerido por NIST. Proporciona confidencialidad e integridad. Este algoritmo se define sólo para entradas que sean múltiplos de 64 bits. La información incluida no tiene por qué ser en realidad una clave. El algoritmo es el mismo independientemente del tamaño de la clave AES utilizada en el empaquetado, denominada clave de cifrado de clave o `KEK`. Los requisitos de implementación se indican a continuación.

Clave de cifrado de clave AES de 128 bits
SE REQUIERE la implementación de envoltura de claves de 128 bits.
Envoltura de otros tamaños de llaves OPCIONAL.
Clave de cifrado de clave AES de 192 bits
Todo el soporte es OPCIONAL.
Clave de cifrado de clave AES de 256 bits
SE REQUIERE la implementación de envoltura de claves de 256 bits.
Envoltura de otros tamaños de llaves OPCIONAL.

Supongamos que los datos que se van a empaquetar constan de `N`bloques de datos de 64 bits denominados `P(1)`, `P(2)`, `P(3)`... `P(N)`. El resultado del ajuste serán `N+1`bloques de 64 bits denominados `C(0)`, `C(1)`, `C(2)`, ... `C(N)`. La clave de cifrado de claves está representada por `K`. Suponga números enteros `i`, `j`y `t`un registro intermedio de 64 bits `A`, un registro de 128 bits `B`y una matriz de cantidades de 64 bits `R(1)`hasta `R(N)`.

"|" representa la concatenación, entonces `x|y`, donde `x` y `y`y cantidades de 64 bits, es la cantidad de 128 bits `x`en los bits más significativos y `y`en los bits menos significativos. `AES(K)enc(x)`es la operación de AES que cifra la cantidad de 128 bits `x`bajo la clave `K`. `AES(K)dec(x)`es la opción de descifrado correspondiente. `XOR(x,y)`es el exclusivo bit a bit o de `x`y `y`. `MSB(x)`y `LSB(y)`son los 64 bits más significativos y los 64 bits menos significativos de xey respectivamente.

Si `N`es 1, se realiza una única operación AES para envolver o desenvolver. Si es `N>1`, entonces `6*N`se realizan operaciones AES para envolver o desenvolver.

El algoritmo de ajuste de claves es el siguiente:

1. Si `N`es `1`:
   - `B=AES(K)enc(0xA6A6A6A6A6A6A6A6|P(1))`
   - `C(0)=MSB(B)`
   - `C(1)=LSB(B)`

   Si es así `N>1`, realice los siguientes pasos:
2. Inicializar variables:
   - Establecer `A`en`0xA6A6A6A6A6A6A6A6`
   - Para , `i=1_N`
     `R(i)=P(i)`
3. Calcular valores intermedios:
   - Para , `j=0_5`
     - Para , `i=1_N`
       `t= i + j*N`
       `B=AES(K)enc(A|R(i))`
       `A=XOR(t,MSB(B))`
       `R(i)=LSB(B)`
4. Salida de los resultados:
   - Colocar`C(0)=A`
   - Para , `i=1_N`
     `C(i)=R(i)`

El algoritmo de desenvolvimiento de claves es el siguiente:

1. Si Nes 1:
   - B=AES(K)dec(C(0)|C(1))
   - P(1)=LSB(B)
   - Si MSB(B)es así 0xA6A6A6A6A6A6A6A6, devuelve el éxito. De lo contrario, devolverá un error de falla de verificación de integridad.

   Si N>1, realice los siguientes pasos:
2. Inicialice las variables:
   - A=C(0)
   - Para , i=1_N
     R(i)=C(i)
3. Calcular valores intermedios:
   - Para , j=5_0
     - Para , i=N_1
       t= i + j*N
       B=AES(K)dec(XOR(t,A)|R(i))
       A=MSB(B)
       R(i)=LSB(B)
4. Salida de los resultados:
   - Para , i=1_N
     P(i)=R(i)
   - Si Aes así 0xA6A6A6A6A6A6A6A6, devuelve el éxito. De lo contrario, devolverá un error de falla de verificación de integridad.

Por ejemplo, envolver los datos 0x00112233445566778899AABBCCDDEEFFcon produce KEK 0x000102030405060708090A0B0C0D0E0Fel texto cifrado de 0x1FA68B0A8112B447,, 0xAEF34BD8FB5A7B82. 0x9D3E862371D2CFE5

5.7 Resumen de mensajes

Los algoritmos de resumen de mensajes se pueden utilizar AgreementMethodcomo parte de la derivación de claves, dentro del cifrado RSA-OAEP como función hash y en conexión con el método del código de autenticación de mensajes HMAC como se describe en [XML-DSIG].)

5.7.1 SHA1

Identificador:
  http://www.w3.org/2000/09/xmldsig#sha1 (REQUERIDO)

El algoritmo SHA-1 [SHA] no toma parámetros explícitos. Un ejemplo de un DigestMethodelemento SHA-1 es:

```
<Método de resumen
 Algoritmo="http://www.w3.org/2000/09/xmldsig#sha1"/>
```

Un resumen SHA-1 es una cadena de 160 bits. El contenido del DigestValueelemento será la codificación base64 de esta cadena de bits vista como un tren de octetos de 20 octetos. Por ejemplo, el DigestValueelemento para el resumen del mensaje:

```
A9993E36 4706816A BA3E2571 7850C26C 9CD0D89D
```

del Apéndice A del estándar SHA-1 sería:

```
<DigestValue>qZk+NkcGgWq6PiVxeFDCbJzQ2J0=</DigestValue>
```

5.7.2 SHA256

Identificador:
  http://www.w3.org/2001/04/xmlenc#sha256 (RECOMENDADO)

El algoritmo SHA-256 [SHA] no toma parámetros explícitos. DigestMethodUn ejemplo de un elemento SHA-256 es:

```
<DigestMethod
 Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
```

Un resumen SHA-256 es una cadena de 256 bits. El contenido del DigestValueelemento será la codificación base64 de esta cadena de bits vista como un tren de octetos de 32 octetos.

5.7.3 SHA512

Identificador:
  http://www.w3.org/2001/04/xmlenc#sha512 (OPCIONAL)

El algoritmo SHA-512 [SHA] no toma parámetros explícitos. DigestMethodUn ejemplo de un elemento SHA-512 es:

```
<Método de resumen
 Algoritmo="http://www.w3.org/2001/04/xmlenc#sha512"/>
```

Un resumen SHA-512 es una cadena de 512 bits. El contenido del DigestValueelemento será la codificación base64 de esta cadena de bits vista como un tren de octetos de 64 octetos.

5.7.4 RIPEMD-160

Identificador:
  http://www.w3.org/2001/04/xmlenc#ripemd160 (OPCIONAL)

El algoritmo RIPEMD-160 [RIPEMD-160] no toma parámetros explícitos. Un ejemplo de un `DigestMethod` elemento RIPEMD-160 es:

```
<Método de resumen
 Algoritmo="http://www.w3.org/2001/04/xmlenc#ripemd160"/>
```

Un resumen RIPEMD-160 es una cadena de 160 bits. El contenido del `DigestValue`elemento será la codificación base64 de esta cadena de bits vista como un tren de octetos de 20 octetos.

5.8 Autenticación de mensajes

Identificador:
http://www.w3.org/2000/09/xmldsig# (RECOMENDADO)

La firma XML [XML-DSIG] es OPCIONAL para implementar en aplicaciones de cifrado XML. Es la forma recomendada de proporcionar autenticación basada en claves.

5.9 Canonicalización

Una canonicalización de XML es un método para serializar XML de forma coherente en un flujo de octetos, según sea necesario antes de cifrar XML.

5.9.1 Canonicalización inclusiva

Identificadores:
http://www.w3.org/TR/2001/REC-xml-c14n-20010315 (OPCIONAL)
http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments (OPCIONAL)

XML canónico [Canon] es un método de serialización de XML que incluye el espacio de nombres dentro del alcance y el contexto del atributo del espacio de nombres xml de los antepasados del XML que se está serializando.

Si XML se va a cifrar y luego descifrar en un entorno diferente y se desea preservar los enlaces de prefijo del espacio de nombres y el valor de los atributos en el espacio de nombres "xml" de su entorno original, entonces se debe utilizar la versión canónica XML con comentarios del XML. Sea la serialización que esté cifrada.

5.9.2 Canonicalización exclusiva

Identificadores:
http://www.w3.org/2001/10/xml-exc-c14n# (OPCIONAL)
http://www.w3.org/2001/10/xml-exc-c14n#WithComments (OPCIONAL)

Canonicalización XML exclusiva [Exclusivo] serializa XML de tal manera que incluye en la medida mínima práctica el enlace de prefijo de espacio de nombres y el contexto de atributo de espacio de nombres xml heredado de elementos ancestros.

Es el método recomendado donde se puede cambiar el contexto externo de un fragmento que fue firmado y luego cifrado. De lo contrario, la validación de la firma sobre el fragmento puede fallar porque la canonicalización mediante validación de firma puede incluir espacios de nombres innecesarios en el fragmento.

6 consideraciones de seguridad

6.1 Relación con las firmas digitales XML

La aplicación de cifrado y firmas digitales en partes de un documento XML puede dificultar el descifrado y la verificación de la firma posteriores. En particular, al verificar una firma se debe saber si la firma se calculó sobre la forma de elementos cifrados o no cifrados.

Un problema aparte, pero importante, es la introducción de vulnerabilidades criptográficas al combinar firmas digitales y cifrado sobre un elemento XML común. Hal Finney ha sugerido que cifrar datos firmados digitalmente, dejando la firma digital en claro, puede permitir ataques de adivinación de texto sin formato. Esta vulnerabilidad se puede mitigar mediante el uso de hashes seguros y nonces en el texto que se procesa.

De acuerdo con el documento de requisitos [EncReq], la interacción entre cifrado y firma es un problema de aplicación y está fuera del alcance de la especificación. Sin embargo, hacemos las siguientes recomendaciones:

1. Cuando los datos están cifrados, cualquier resumen o firma sobre esos datos debe cifrarse. Esto satisface el primer problema en el sentido de que sólo se pueden validar aquellas firmas que se pueden ver. También aborda la posibilidad de una vulnerabilidad de adivinación de texto sin formato, aunque puede que no sea posible identificar (o incluso conocer) todas las firmas de un determinado dato.
2. Emplee la transformación de firma "decrypt-except" [XML-DSIG-Decrypt] . Funciona de la siguiente manera: durante el procesamiento de transformación de firma, si encuentra una transformación de descifrado, descifre todo el contenido cifrado del documento excepto aquellos exceptuados por un conjunto enumerado de referencias.

Además, si bien las siguientes advertencias se refieren a inferencias incorrectas por parte del usuario sobre la autenticidad de la información cifrada, las aplicaciones deben disuadir la mala interpretación del usuario comunicando claramente qué información tiene integridad o está autenticada, es confidencial o no repudiable cuando se realizan múltiples procesos (por ejemplo, firma). y cifrado) y se utilizan algoritmos (por ejemplo, simétricos y asimétricos):

1. Cuando un sobre cifrado contiene una firma, la firma no necesariamente protege la autenticidad o integridad del texto cifrado [Davis].
2. Si bien la firma protege el texto sin formato, solo cubre lo que está firmado, los destinatarios de los mensajes cifrados no deben inferir la integridad o autenticidad de otra información sin firmar (por ejemplo, encabezados) dentro del sobre cifrado; consulte [XML-DSIG , 8.1.1 Solo lo que está firmado es seguro].

6.2 Información revelada

Cuando una clave simétrica se comparte entre varios destinatarios, esa clave simétrica *solo* debe usarse para los datos destinados a *todos* los destinatarios; Incluso si un destinatario no es dirigido a información destinada (exclusivamente) a otro en la misma clave simétrica, la información podría ser descubierta y descifrada.

Además, los diseñadores de aplicaciones deben tener cuidado de no revelar ninguna información en los parámetros o identificadores de algoritmos (por ejemplo, información en un URI) que debilite el cifrado.

6.3 Nonce y IV (Valor o Vector de Inicialización)

Una característica indeseable de muchos algoritmos de cifrado y/o sus modos es que el mismo texto sin formato, cuando se cifra con la misma clave, tiene el mismo texto cifrado resultante. Si bien esto no es sorprendente, invita a varios ataques que se mitigan al incluir datos arbitrarios y no repetidos (bajo una clave determinada) con el texto sin formato antes del cifrado. En los modos de encadenamiento de cifrado, estos datos son los primeros en cifrarse y, en consecuencia, se denominan IV (valor de inicialización o vector).

Los diferentes algoritmos y modos tienen requisitos adicionales sobre las características de esta información (por ejemplo, aleatoriedad y secreto) que afectan las características (por ejemplo, confidencialidad e integridad) y su resistencia a los ataques.

Dado que los datos XML son redundantes (por ejemplo, codificaciones Unicode y etiquetas repetidas) y que los atacantes pueden conocer la estructura de los datos (por ejemplo, DTD y esquemas), los algoritmos de cifrado deben implementarse y utilizarse cuidadosamente en este sentido.

Para el modo Cipher Block Chaining (CBC) utilizado por esta especificación, el IV no debe reutilizarse para ninguna clave y debe ser aleatorio, pero no es necesario que sea secreto. Además, en este modo, un adversario que modifique el IV puede realizar un cambio conocido en el texto sin formato después del descifrado. Este ataque se puede evitar asegurando la integridad de los datos de texto sin formato, por ejemplo firmándolos.

6.4 Denegación de Servicio

This specification permits recursive processing. For example, the following scenario is possible: `EncryptedKey` **A** requires `EncryptedKey` **B** to be decrypted, which itself requires `EncryptedKey` **A**! Or, an attacker might submit an `EncryptedData` for decryption that references network resources that are very large or continually redirected. Consequently, implementations should be able to restrict arbitrary recursion and the total amount of processing and networking resources a request can consume.

6.5 Unsafe Content

XML Encryption can be used to obscure, via encryption, content that applications (e.g., firewalls, virus detectors, etc.) consider unsafe (e.g., executable code, viruses, etc.). Consequently, such applications must consider encrypted content to be as unsafe as the unsafest content transported in its application context. Consequently, such applications may choose to (1) disallow such content, (2) require access to the decrypted form for inspection, or (3) ensure that arbitrary content can be safely processed by receiving applications.

7 Conformance

An implementation is conformant to this specification if it successfully generates syntax according to the schema definitions and satisfies all MUST/REQUIRED/SHALL requirements, including algorithm support and processing. Processing requirements are specified over the roles of ***decryptor***, ***encryptor***, and their calling ***application***.

8 Tipo de medio de cifrado XML

8.1 Introducción

Sintaxis y procesamiento de cifrado XML [XML-Encryption] especifica un proceso para cifrar datos y representar el resultado en XML. Los datos pueden ser datos arbitrarios (incluido un documento XML), un elemento XML o contenido de un elemento XML. El resultado del cifrado de datos es un elemento de cifrado XML que contiene o hace referencia a los datos cifrados.

El `application/xenc+xml`tipo de medio permite que las aplicaciones de cifrado XML identifiquen documentos cifrados. Además, permite que las aplicaciones que conocen este tipo de medio (incluso si no son implementaciones de cifrado XML) tengan en cuenta que el tipo de medio del objeto descifrado (original) puede ser un tipo distinto de XML.

8.2 aplicación/xenc+xml Registro

Este es un registro de tipo de medio tal como se define en Extensiones multipropósito de correo de Internet (MIME), parte cuatro: Procedimientos de registro [MIME-REG]

Nombre del tipo de medio MIME: aplicación

Nombre del subtipo MIME: xenc+xml

Parámetros requeridos: ninguno

Parámetros opcionales: juego de caracteres

> Los valores permitidos y recomendados y la interpretación del parámetro charset son idénticos a los proporcionados para 'application/xml' en la sección 3.2 de RFC 3023 [XML-MT].

Consideraciones de codificación:

> Las consideraciones de codificación son idénticas a las dadas para 'aplicación/xml' en la sección 3.2 de RFC 3023 [XML-MT].

Consideraciones de Seguridad:

> Consulte la sección Consideraciones de seguridad [Cifrado XML] .

Consideraciones de interoperabilidad: ninguna

Especificación publicada: [Cifrado XML]

Aplicaciones que utilizan este tipo de medio:

XML Encryption es neutral en cuanto a dispositivos, plataformas y proveedores y es compatible con una variedad de aplicaciones web.

Información adicional:

Número(s) mágico(s): ninguno

Aunque no se puede contar con secuencias de bytes para identificar consistentemente los documentos XML Encryption, serán documentos XML en los que el elemento raíz ' `QName`s `LocalPart`es `'EncryptedData'`o ' `EncryptedKey`' con un nombre de espacio de nombres asociado de ' http://www.w3.org/2001/ 04/xmlenc# '. El `application/xenc+xml`nombre del tipo DEBE usarse solo para objetos de datos en los que el elemento raíz sea del espacio de nombres de cifrado XML. Los documentos XML que contienen estos tipos de elementos en lugares distintos del elemento raíz se pueden describir utilizando funciones como [esquema XML].

Extensiones de archivo: .xml

Código(s) de tipo de archivo de Macintosh: "TEXTO"

Persona y dirección de correo electrónico a contactar para obtener más información:

José Reagle <reagle@w3.org>

Grupo de trabajo XENC <xml-encryption@w3.org>

Uso previsto: COMÚN

Autor/Cambiar controlador:

La especificación de cifrado XML es un producto del trabajo del World Wide Web Consortium (W3C), que tiene control de cambios sobre la especificación.

9 esquema y ejemplos válidos

Esquema
esquema-xenc.xsd
Ejemplo
enc-example.xml (no es criptográficamente válido pero ejercita gran parte del esquema)

10 referencias

TRIPLES
ANSI X9.52: Modos de operación del algoritmo de cifrado de datos triple. 1998.
AES
NIST FIPS 197: Estándar de cifrado avanzado (AES) . Noviembre de 2001.
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
ENVOLTURA AES
RFC3394: Algoritmo de ajuste de claves del estándar de cifrado avanzado (AES) . J. Schaad y R. Housley. Informativo, septiembre de 2002.
Algoritmos CMS
RFC3370: Algoritmos de sintaxis de mensajes criptográficos (CMS) . R. Housley. Informativo, febrero de 2002.
http://www.ietf.org/rfc/rfc3370.txt
Envoltura CMS
RFC3217: Encapsulación de claves Triple-DES y RC2 . R. Housley. Informativo, diciembre de 2001.
http://www.ietf.org/rfc/rfc3217.txt
davis
Firma y cifrado defectuosos en S/MIME, PKCS#7, MOSS, PEM, PGP y XML. D. Davis. Conferencia Técnica Anual de USENIX. 2001.
http://www.usenix.org/publications/library/proceedings/usenix01/davis.html
DES
NIST FIPS 46-3: Estándar de cifrado de datos (DES). Octubre de 1999.
http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf
EncReq
Requisitos de cifrado XML . J. Reagle. Nota del W3C, marzo de 2002.
http://www.w3.org/TR/2002/NOTE-xml-encryption-req-20020304
ESDH
RFC 2631: Método de acuerdo de claves Diffie-Hellman. E. Rescorla. Seguimiento de estándares, 1999.
http://www.ietf.org/rfc/rfc2631.txt
http://www.w3.org/TR/2002/CR-xml-exc-c14n-20020212
Glosario
RFC 2828: Glosario de seguridad de Internet . R Shirey. Informativo, mayo de 2000.
http://www.ietf.org/rfc/rfc2828.txt
HMAC
RFC 2104: HMAC: hash con clave para autenticación de mensajes . H. Krawczyk, M. Bellare y R. Canetti. Informativo, febrero de 1997.
http://www.ietf.org/rfc/rfc2104.txt
HTTP
RFC 2616: Protocolo de transferencia de hipertexto - HTTP/1.1. J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach y T. Berners-Lee. Standards Track, junio de 1999.
http://www.ietf.org/rfc/rfc2616.txt
PALABRAS CLAVE
RFC 2119: Palabras clave para uso en RFC para indicar niveles de requisitos. S. Bradner. Mejores prácticas actuales, marzo de 1997.

http://www.ietf.org/rfc/rfc2119.txt
MD5
RFC 1321: Algoritmo de resumen de mensajes MD5. R. Rivest. Informativo, abril de 1992.
http://www.ietf.org/rfc/rfc1321.txt
MÍMICA
RFC 2045: Extensiones multipropósito de correo de Internet (MIME), primera parte: Formato de los cuerpos de los mensajes de Internet . N. Freed y N. Borenstein. Standards Track, noviembre de 1996.
http://www.ietf.org/rfc/rfc2045.txt
MIME-REG
RFC 2048: Extensiones multipropósito de correo de Internet (MIME), cuarta parte: Procedimientos de registro . N. Freed, J. Klensin y J. Postel. Mejores prácticas actuales, noviembre de 1996.
http://www.ietf.org/rfc/rfc2048.txt
NFC
TR15, formularios de normalización Unicode . M. Davis y M. Dürst. Revisión 18: noviembre de 1999.
http://www.unicode.org/unicode/reports/tr15/tr15-18.html .
Corrección NFC
"Corrigéndum n.º 2: Yod con normalización de Hiriq ".
http://www.unicode.org/versions/corrigendum2.html .
prop1
"Propuesta de hombre de paja de cifrado XML ". E. Simon y B. LaMacchia. Agosto de 2000.
http://lists.w3.org/Archives/Public/xml-encryption/2000Aug/0001.html
prop2
Otra propuesta de Cifrado XML . T. Imamura. Agosto de 2000.
http://lists.w3.org/Archives/Public/xml-encryption/2000Aug/0005.html
prop3
Sintaxis y procesamiento de cifrado XML . B. Dillaway, B. Fox, T. Imamura, B. LaMacchia, H. Maruyama, J. Schaad y E. Simon. Diciembre de 2000.
http://lists.w3.org/Archives/Public/xml-encryption/2000Dec/att-0024/01-XMLEncryption_v01.html
PKCS1
RFC 2437: PKCS #1: Especificaciones de criptografía RSA Versión 2.0. B. Kaliski y J. Staddon. Informativo, octubre de 1998.
http://www.ietf.org/rfc/rfc2437.txt
ALEATORIO
RFC 1750: Recomendaciones de aleatoriedad para la seguridad . D. Eastlake, S. Crocker y J. Schiller. Informativo, diciembre de 1994.
http://www.ietf.org/rfc/rfc1750.txt
RIPEMD-160
CryptoBytes, Volumen 3, Número 2. La función hash criptográfica RIPEMD-160 . Laboratorios RSA. Otoño de 1997.
ftp://ftp.rsasecurity.com/pub/cryptobytes/crypto3n2.pdf
http://www.esat.kuleuven.ac.be/~cosicart/pdf/AB-9601/AB-9601.pdf
sha
Estándar de hash seguro. NIST FIPS 180-1. (RFC 3174). Abril de 1995.
http://www.itl.nist.gov/fipspubs/fip180-1.htm
Estándar de hash seguro. Borrador NIST FIPS 180-2 . 2001. (Ampliado para incluir SHA-384, SHA-256 y SHA-512)
http://csrc.nist.gov/encryption/shs/dfips-180-2.pdf
A Bin
R. Tobin. Conjunto de información para entidades externas , lista de correo XML Core, 2000 [solo miembro del W3C].
http://lists.w3.org/Archives/Member/w3c-xml-core-wg/2000OctDec/0054
UTF-16
RFC 2781: UTF-16, una codificación de ISO 10646. P. Hoffman y F. Yergeau. Informativo, febrero de 2000.
http://www.ietf.org/rfc/rfc2781.txt
UTF-8
RFC 2279: UTF-8, un formato de transformación de ISO 10646F.F. Yergeau. Standards Track, enero de 1998.
http://www.ietf.org/rfc/rfc2279.txt
URI
RFC 2396: Identificadores uniformes de recursos (URI): sintaxis genérica . T. Berners-Lee, R. Fielding y L. Masinter. Standards Track, agosto de 1998.
http://www.ietf.org/rfc/rfc2396.txt
http://www.ietf.org/rfc/rfc1738.txt
http://www.ietf.org/rfc/rfc2141.txt
RFC 2611: Mecanismos de definición de espacios de nombres URN.Mejores prácticas actuales. Daigle, D. van Gulik, R. Iannella, P. Falstrom. Junio de 1999.
http://www.ietf.org/rfc/rfc2611.txt
X509v3
Recomendación UIT-T X.509 versión 3 (1997). "Tecnología de la información - Interconexión de sistemas abiertos - El marco de autenticación de directorios" ISO/IEC 9594-8:1997.
XML
Lenguaje de marcado extensible (XML) 1.0 (segunda edición) . T. Bray, J. Paoli, CM Sperberg-McQueen y E. Maler. Recomendación del W3C, octubre de 2000.
Base XML
Base XML . J. Marsh. Recomendación del W3C, junio de 2001.
http://www.w3.org/TR/2001/REC-xmlbase-20010627/
XML-C14N
XML canónico.J. Boyer. Recomendación del W3C, marzo de 2001.
http://www.w3.org/TR/2001/REC-xml-c14n-20010315
http://www.ietf.org/rfc/rfc3076.txt
XML-exc-C14N
Canonicalización XML exclusiva . J. Boyer, D. Eastlake y J. Reagle. Recomendación del W3C, julio de 2002.
http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/
XML-DSIG
Sintaxis y procesamiento de firmas XML . D. Eastlake, J. Reagle y D. Solo. Recomendación del W3C, febrero de 2002.
http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/
XML-DSIG-Descifrar
Transformación de descifrado para firma XML . M. Hughes, T. Imamura y H. Maruyama. Recomendación del W3C, diciembre de 2002.

http://www.w3.org/TR/2002/REC-xmlenc-decrypt-20021210
Cifrado XML
Sintaxis y procesamiento de cifrado XML . D. Eastlake y J. Reagle. Recomendación candidata del W3C, diciembre de 2002. http://www.w3.org/TR/2002/CR-xmlenc-core-20020802/
Conjunto de información XML
Conjunto de información XML . J. Cowan y R. Tobin. Recomendación del W3C, octubre de 2001 http://www.w3.org/TR/2001/REC-xml-infoset-20011024/
XML-MT
RFC 3023: tipos de medios XML. M. Murata, S. St. Laurent y D. Kohn. Informativo, enero de 2001. http://www.ietf.org/rfc/rfc2376.txt
XML-NS
Espacios de nombres en XML . T. Bray, D. Hollander y A. Layman. Recomendación del W3C, enero de 1999. http://www.w3.org/TR/1999/REC-xml-names-19990114
esquema XML
Esquema XML Parte 1: Estructuras D. Beech, M. Maloney y N. Mendelsohn. Recomendación del W3C, mayo de 2001. http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/
Esquema XML, parte 2: tipos de datos . P. Biron y A. Malhotra. Recomendación del W3C, mayo de 2001. http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/
XPath
Lenguaje de ruta XML (XPath) versión 1.0 . J. Clark y S. DeRose. Recomendación del W3C, octubre de 1999. http://www.w3.org/TR/1999/REC-xpath-19991116